

- **OS/2 Performance Primer**
- **Interactive PM Programming**

January/February 1995
Display Unit 2/28/1995

PRODUCT REVIEW

VX-REXX
Client/Server
Edition

OS/2 Developer

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

**PERFORMANCE
TESTING**

Plugging Memory Leaks

**Multiple Platform
Development**

**Exception Handlers That
Should Be The Rule**

**A User-Customizable
Toolbar Control**

**BUYERS GUIDE:
Testing and
Debugging Tools**

**OS/2 CODE
THAT**

**MAKES THE
GRADE**

U.S. \$9.95 Vol. 7 No. 1



A Miller Freeman Publication

Take the
"black holes"
out of
your test
coverage.

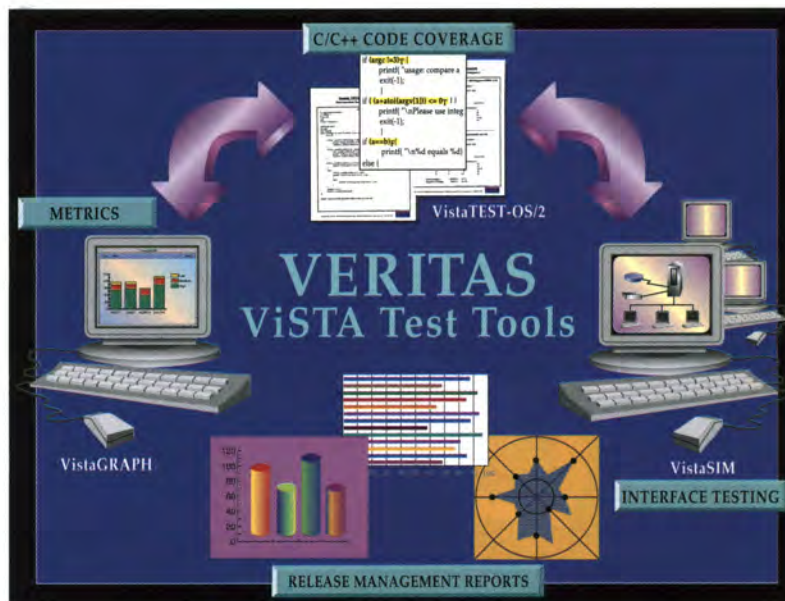


ViSTA™

Before Your
Clients Start
Disappearing...

Black holes may be an interesting phenomenon in outer space, but not in your software testing. We have helped 100's of companies like yours, on over 200 million lines of code, verify an amazing fact: less than 40% of "production level" code is being tested. Now that is a major "black hole".

... Into the
Unknown.



A POWERFUL SET OF ENTERPRISE TESTING TOOLS

VistaTEST-OS/2

- C/C++
- Multi platform
- Unit level testing
- Complexity Metrics
- Call Coverage, Multiconditional, Branch, Path, Function, Interface
- DDL/Multithreaded 16bit/32bit support

Code Coverage

VistaSIM

- Client/Server Testing
- Reuse Testing
- Automate Error Handling Test
- Over 1200 OS/2 Sim stubs for OS calls, PM calls, ANSI

Interface Testing

VistaGRAPH

- Metric summary
- Flexible Spreadsheet (diagrams and graphs)
- Quantifies testing process
- Project team oriented
- ISO 9000

Release Mgmt.

UNIX POWER TOOLS NOW ON OS/2

IBM	HP	INTEL
- AIX TEAM	ADP	DIGITAL
- OS/2 TEAM	APPLE	UNISYS
- WPOS TEAM	ARBOR	CITIBANK
MOTOROLA	OSF	MEAD
NOVELL	SUN	DATA CARD

ViSTA tools help increase the quality of your application, isolate dead code and improve your client/server testing.



Call 1-800-258-8649 today for a
Free Evaluation.

FAX at 408-562-4334 or e-mail vsales@veritas.com

Be sure to ask about our
OS/2 Enterprise Workgroup
for extra savings.



VERITAS ViSTA, VistaSIM, VistaGRAPH and VistaTEST are trademarks of VERITAS Software.



TechBridge Builder
Version 1.1 for OS/2
NOW AVAILABLE
Introductory Offer until Jan 31, 1995

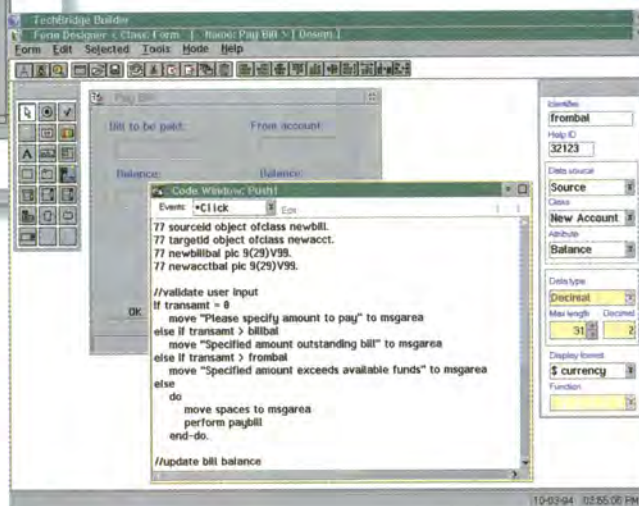
Introductory Offer

• Enterprise Edition \$1,650.00

• Bronze Edition **\$ 750.00**
(Local DB2/2 access Only)

Regular Price \$2,850.00

Royalty-free runtime



TechBridge makes Client/Server Development *Easy*

Your application is only a few mouse clicks away

Easy POINT-AND-CLICK TIES GUI FIELDS TO DATA

Automatically connect GUI fields to your database, and you have a form that can view and update data. Or import table definitions and let us auto-design a quick form for you to do query or present the data in 3-D business charts. Access DB2/2 and the whole DB2 family through DDCS/2; or Oracle 7 or 50 more data sources through EDA/SQL middleware. With **TechBridge Builder** you can do all this, with just drag-and-drop, point-and-click and absolutely **NO CODING** because so much is **AUTOMATED**.

Easy VISUAL DEVELOPMENT

Place a pre-defined control group from your reuse library and attach business logic or context sensitive help. Double-click to preview. Design a modal dialog or an advanced OOUI workplace with point-and-click. Use drag-and-drop to tie your objects together into an application, then let our packager produce the executables for you.

Easy SCRIPTING

Double click on a form control and enter an event driven script, e.g. Multiply Principle by Interest giving Amount. Yes, it's just COBOL. Click to compile it. Click again to test it. Click to invoke the debugger. Iterative rapid development? You bet!

Easy OBJECT TECHNOLOGY

With **TechBridge Builder** you get a complete set of classes, the practical building blocks that you can tailor to build your application. Supporting these classes is a framework with over 700 base classes and 24,000 methods - a knowledge base of technology like CUA '91, SQL, PM, and human factors. Don't waste time crafting your own framework, build on ours and jump right into building applications that meet your specific needs.

Easy ON YOUR SYSTEM

The minimum system requirement is an 8M OS/2 machine with 10M of free disk space. It's LAN enabled.



1-800-463-8998



**TechBridge
Technology Corp.**

TechBridge Technology Corp. 5001 Yonge St., Suite 1301, North York, Ontario, Canada M2N 6P6 Phone: (416) 222-8998 Fax: (416) 222-0168
Prices and specification are subject to change without notice. Price does not include freight and taxes where applicable. Prices quoted in US dollars. TechBridge, TechBridge Technology and TechBridge Builder are registered trademarks of TechBridge Technology Corp. ©1994 TechBridge Technology Corp. All other brand and product names are trademarks or registered trademarks of their respective companies.

Circle Reader Service Number 2



With so many advanced features in Micro Focus COBOL, there's one you may have overlooked.

The Future.

At Micro Focus, we have a history of ensuring a bright future for our customers.

Fifteen years ago, when the "future" was personal computers, we brought the first true business application development environment to the desktop. Over the years, we built on that foundation by providing the tools and utilities that delivered new levels of productivity to programmers.

Today, we're ready for the next step: Introducing Object COBOL™—the first true object oriented business programming environment.

The Micro Focus Object COBOL Option provides all the functionality you would expect from

an object-oriented development environment—including encapsulation, polymorphism and inheritance. It also brings all the benefits of object orientation—reusability, real-world modeling and increased maintainability—to COBOL programmers without discarding existing investments in code and skills.

Object COBOL is shipped with a library of classes for managing collections of objects and for creating Graphical User Interfaces. These classes can be accessed and extended with another Object COBOL component, the Browser.

Object COBOL even extends the COBOL language by letting you define syntax that best

suits your business needs. Object COBOL's unique vocabularies make applications easier to read, write and understand... for programmers and end-users alike!

And the best part? It's designed for COBOL programmers, so with Micro Focus, if you know COBOL, you're ready for an object-oriented future today.

For the latest update on this new technology, call 800-MF-COBOL and ask for our white paper: The Object Oriented COBOL Model.

Micro Focus: The past, present, and future of programming.

MICRO FOCUS®

Micro Focus is a registered trademark of Micro Focus Ltd. Object COBOL is a trademark of Micro Focus.

Circle Reader Service Number 3

OS/2 Developer

JANUARY/FEBRUARY 1995

THE MAGAZINE FOR ADVANCED SOFTWARE DEVELOPMENT

```
#define INCL_DOS
#define INCL_DOSMEMMGR
#include <os2.h>
#include <malloc.h>
#include <sememf.h>
#include <process.h>
#include <stdio.h>
#include <stdlib.h>

int main( int argc, char *argv )
{
    PVOID memadd=NULL;
    ULONG size=40000;
    PCHAR help=NULL;
    int i=0;
    /* Do nothing, but just consuming "
    i=0;
    for (i=1;i<1000;i++) {
        DosAllocMem(&memadd,size,f
        ALLOC); help =
        (PCHAR)memadd;
        memset(memadd,'\0',size);
        strcpy(help,"I'm a memory eater"
        DosSleep(500);
    } /* endfor */
    return;
```



EDITOR'S COMMENTS

More Talent, Tools, and Tips 5



OS/2 TOOLBOX

Product Reviewed: VX-REXX Client/Server Edition 8



GUI CORNER

An Exception You Can Handle 16



PROGRAMMING INSIDER

An API by Any Other Name... 22



PERFORMANCE, QUALITY ASSURANCE

Advanced Technology Needs Advanced Testing 26

By Joseph P. Busa

Plug Those Memory Leaks! 40

By Steve Hargis and Mike Skelton

An OS/2 Performance Primer 65

By Dieter Babutzka



SOFTWARE TOOLS

Multiple Platform Development 31

By Bruce Landeck

Interactive PM Programming 70

By David Liebttag

Testing and Debugging Tools Buyer's Guide 78



GUI

A User-Customizable Toolbar Control 50

By Mark McMillan, Gennaro Cuomo, Jürg von Känel, and Guillaume Le Stum



PRODUCT WATCH

New Tools for OS/2 76





Programmer's Paradise®

Call for a
FREE
Catalog!



Gammatech REXX SuperSet/2 by Softouch Systems

The SuperSet/2 gives REXX the power of C by providing interfaces to LAN Server, NetBIOS, TCP/IP and Communications Manager/2, and by providing access to low level system facilities, including processes, threads, semaphores, names pipes, and VIO commands. The SuperSet/2 complements visual REXX programming packages. The 600+ page manual fully documents over 300 functions in 7 DLLs ready to supercharge your REXX!

List: \$80 Ours: \$69 FAXcetera #: 3621-0002

key 01JF95

Visual SlickEdit™ for OS/2 by MicroEdge

The same great programmers' editor that gave you extensive language support, project management, and superb editing features is now available on OS/2! Users of Visual SlickEdit for Windows or NT don't have to rewrite their macros when switching to OS/2. Macro source, dialog templates, and bitmaps are binary-compatible with Visual SlickEdit on all other platforms. 30 day risk-free trial!

List: \$295 Ours: \$199 FAXcetera #: 1997-0002



key 03JF95



IBM® OS/2® Warp v3 by IBM

OS/2 Warp Version 3, the reliable 32-bit operating system, makes computing easier. Features include a launch pad for easy access to your frequently used applications, built-in multimedia support, 3D animated icons and one-button install. Every copy includes a BonusPak for no extra charge. This BonusPak provides you with a complete set of tools for accessing and navigating the Internet; IBM Works—a suite of 32-bit applications; FaxWorks for OS/2; HyperACCESS Lite; IBM Person to Person®; and much more. Systems with 4MB of memory are now supported.

List: \$129 Ours: \$79 FAXcetera #: 3142-0035

key 05JF95

EZRAID by PRO Engineering, Inc.

EZRAID PRO is the OS/2 software solution that provides the same full powered RAID benefits found in hardware systems worldwide. With its unique technology, EZRAID PRO increases system performance, simplifies data management and ensures complete fault tolerance. EZRAID PRO supports spanning, RAID levels 0, 1, 4 and 5; is compatible with SCSI, IDE and ESDI; and is Ready for Warp and LAN Server 4.0.

Lite List: \$195 Ours: \$188
PRO List: \$795 Ours: \$764
FAXcetera #: 1016-5101



key 07JF95

RimStar™ Programmer's Editor by RimStar Technology, Inc.



Features: 32-bit GUI, Syntax Coloring, ANSI "C" macro language, "C" Source Browser, unlimited redo/undo, no limits on number of files, windows, or line length. Emulates BRIEF plus many other editors. Compile/jump to error, hex editing, smart indenting, bookmarks, completely configurable keyboard mappings, WF/2 enabled and much more. Window & NT version available.

List: \$299 Ours: \$259 FAXcetera #: 1013-0901

key 02JF95

The Object Factory—IDL by Synaptec, Inc.

IDL development environment for OS/2 SOM. Includes multiple inheritance, framework filters, Drag drop backup and restore, a SOM class browser, on-line help and much more. Use class development notebooks to create new classes just by dropping them on the class browser. Eliminate the complexities of SOM programming. Framework filters focus development providing quicker access to the hundreds of SOM classes. View every inherited method from every parent in one list box. Manage class development with great ease and little effort.

List: \$250 Ours: \$229 FAXcetera #: 1012-5402

key 04JF95



c-tree Plus® by FairCom

DOS • WINDOWS • NT • UNIX • OS/2 • SUN • RS6000 • HP9000 • MAC • QNX • BANYAN • SCO. This well known, highly portable data management package has become established as the tool of choice for commercial development. Offering unprecedented data control, choose from direct low level access, ISAM level, or SQL access with the FairCom Server. Single User, MultiUser, or optional Client/Server, ANSI Standard. Full Source. No Royalties.

List: \$595 Ours: \$495 FAXcetera #: 1381-0004

key 06JF95



To order call: 800-445-7899
Corporate Developer Division: 800 441-1511
FAX: 908 389-9227
International: 908 389-9228
Customer Service: 908 389-9229
Programmer's Paradise Deutschland:
Tel.: 08121/79073 • FAX: 08121/76566
Programmer's Paradise Italia:
Tel.: 39-2-967-00409 • FAX: 39-2-967-02855
Programmer's Paradise United Kingdom:
Tel.: 0161 7284177 • FAX: 0161 7284017
For more information on the
products featured on these pages call
FAXcetera®: (908) 389-8173

Programmer's Paradise®

1163 Shrewsbury Avenue
Shrewsbury, NJ 07702-4321

Circle Reader Service Number 4

• Prices subject to change without notice.
• Call for shipping charges/return policies.

9
8
7
6
5
4
3
2
1
0
0
0
8

EDITOR*Dick Conklin***EXECUTIVE EDITOR***Nicole Freeman***MANAGING EDITOR***Marta Dils***EDITORIAL ASSISTANT***Diane Anderson***EDITORIAL ARTIST***Peter Altenberg***VICE PRESIDENT/****GROUP DIRECTOR***Regina Starr Ridley***PUBLISHER***Peter Westerman***ADVERTISING SALES***Yvonne Labat**(415) 905-2353**Holly Meintzer**(212) 626-2275**Kristin Morgan**(212) 626-2498***MARKETING MANAGER***Susan McDonald***ART DIRECTOR/MARKETING***Christopher H. Clarke***MARKETING ASSISTANT***Larra Dutton***ADVERTISING PRODUCTION****COORDINATOR***Glenn Sadin***CIRCULATION DIRECTOR***John Rockwell***CIRCULATION MANAGER***Lucinda Formyduval***SUBSCRIPTION INQUIRIES***U.S.: (800) WANT-OS2**Foreign: (708) 647-5960***CHAIRMAN OF THE BOARD***Graham J.S. Wilson***PRESIDENT/CEO***Marshall W. Freeman***EXECUTIVE VICE PRESIDENT***Thomas L. Kemp***VICE PRESIDENT, PRODUCTION***Andy Mickus***VICE PRESIDENT, CIRCULATION***Jerry Okabe***BY DICK CONKLIN**

Editor's Comments

More Talent, Tools, and Tips

What happened to 1994? It's hard to believe that another year is already starting. This is an appropriate time to welcome two new additions to OS/2 Developer: Peter Westerman and Guy Scharf.

Peter is our new Publisher, replacing the irreplaceable Cathy Passage. Peter comes to us from *Microsoft Systems Journal*, of all places, where he will still maintain the duties of publisher. *Mathematica* completes his triad of magazines. Welcome to the other side, Peter!

Guy is another welcome addition to our pages. We have been without a software tools reviewer; now that slot is filled. Guy is no stranger to OS/2 programming. He's the sysop of the OS/2 Vendor and OS/2 Shareware forums on CompuServe. He is also the cofounder, treasurer, and newsletter editor for the OS/2 Bay Area Users Group. Guy will be reviewing the latest and greatest tools for

OS/2 application development. If you have a favorite tool to recommend, or one that you just want to know more about, let Guy know.

RACY PICTURES

The second annual OS/2 Developer 5K Race took place during Software Development '95 East. Those of you who participated were treated to an invigorating, early morning tour of our nation's capital. If you didn't stop to sightsee along the route, you just may have been a winner.

*Dick Conklin*

TALK TO US!

In 1995, as in 1994, we are striving to make OS/2 Developer a publication that works for you. We have made some changes and plan to make more, but you are in the driver's seat. What kind of articles do you like most? Are you seeking more examples, tips, and hints? Are the code listings helpful? Do you use the sample program files in our CompuServe OS2DF2 forum? Have we overlooked any topics? Are we overdoing any? Take a moment and e-mail us with your comments; we want to hear from you!

Have a Great 1995!

Dick Conklin



OS/2 Developer: Vol. 7, No. 1, January/February 1995

Subscription information: U.S.: One year (six issues), \$39.95. Canada, Mexico, and international surface mail, add \$16 per year for postage. Canadian GST no. 124513185. International air mail, add \$30 per year for postage. Foreign orders must be accompanied by payments in U.S. funds. For new orders and customer service, call (800) WANT-OS2 (800-926-8672) or (708) 647-5960 (fax: 708-647-5972). IBM employees and branch office customers can subscribe to OS/2 Developer through IBM Mechanicsburg's Systems Library Services (SLSS) using OS/2 Developer's order number: G362-0001. POSTMASTER: Please send address changes to OS/2 Developer, P.O. Box 1079, Skokie, IL 60076-8079. Allow eight weeks for subscriptions to begin. OS/2 Developer (ISSN 1073-0729) is published bimonthly by Miller Freeman Inc., 600 Harrison Street, San Francisco, CA 94107, (415) 905-2200. Application to mail at second-class postage rates is pending at San Francisco, CA, and additional mailing offices.

OS/2 Developer has applied for BPA membership.

© Copyright 1995 by Miller Freeman Inc. PRINTED IN THE U.S.A.

un

mf Miller Freeman, Inc.
A MEMBER OF THE UNITED NEWSPAPERS GROUP

A Special Report from the publishers of OS/2 Developer!

Quantities are limited.
Order now!
For fastest service,
fax your order to us
at 415-905-4967.



Smalltalk



☒ **YES!** Please send me _____ copies of Smalltalk!

Each copy is only \$9.95 in the US or \$12.95 in Canada.

Please add \$2.50 for shipping and handling.

- ☐ My check is enclosed
- ☐ Charge my credit card: ☐ Visa ☐ MasterCard ☐ American Express

Name

Signature

Date

Send them to:

Name

Address

City

State

Postal Code

Mail: Smalltalk
P. O. Box 7046
San Francisco
CA 94120

Phone: 1-800-444-4881

Fax: (415) 905-4967

You Discovered the 32-Bit Power of OS/2.

FOR CORPORATE
DEVELOPERS BUILDING
SOLUTIONS ON
OS/2!

Now Master It at OS/2 DEVELOPMENT '95.



Introducing the conference for advanced software development.



The publishers of OS/2 Developer magazine have assembled industry leaders to host the first independent Pan-European conference on OS/2. You'll sharpen your OS/2 skills and learn from the experts about the latest techniques and technologies to increase your effectiveness on the job.

What's included:

- Excellent educational seminars
- Keynote speakers
- Product Roundtables
- And special events!



You'll Learn More About

- SOM programming
- Workplace shell programming
- User interface development
- OpenDOC
- Object-oriented programming
- REXX programming
- SmallTalk, and more!

Produced by

**OS/2
Developer**
magazine

May 8-10, 1995
Krasnapolsky Hotel
Amsterdam

To find out more
about this exciting
OS/2 event, fax,
phone, write, or
contact us by
electronic mail
now!

Telephone:
32-2-358.60.40
From the US:
011-32-2-358.60.40

CompuServe:
76307,1054

Internet:
pwesterman@mfi.com

Mail:
OS/2 Development '95
Chaussée de Charleroi 123a
Bte 5
B-1060 Brussels, Belgium

Miller Freeman, Inc.
A MEMBER OF THE UNITED NEWSWEEK GROUP



DEVELOPMENT '95

**SEND BACK THIS
COUPON TO
FAX 32-2-537.56.26
FROM THE US:
011-32-2-537.56.26**

Name

Title Company

Mailing Address:

Telephone

Fax Email



Our new OS/2 Toolbox column will review products that help develop applications for the OS/2 platform.

By GUY SCHARF

Product Reviewed: VX-REXX Client/ Server Edition



Guy Scharf

Welcome to the OS/2 Toolbox. In this column, I will be reviewing tools for the professional OS/2 developer. I am especially interested in products that help in developing high-quality Presentation Manager applications. I'll tell you what I see as the strong points of each tool as well as point out the limits of applicability and any major warts.

VX-REXX

VX-REXX for OS/2 is a powerful yet easy-to-use visual application development tool that uses REXX as its programming language. Watcom has just released VX-REXX for OS/2 Version 2.1 and a new product named VX-REXX for OS/2 Client/Server Edition. The Client/Server Edition is version 2.1 with database and charting functions added. I tested the Client/Server Edition and found it a pleasure to use.

VX-REXX helps you create native OS/2 Presentation Manager applications. The integrated development environment includes a debugger, graphical design tools, and drag-and-drop programming. VX-REXX supports OS/2 Presentation Manager controls, including the spin button, container, notebook, value set, and slider. VX-REXX adds picture radio and push-button controls.

Timer and DDEClient objects support timer events and building a DDE client application.

When you are creating an application, VX-REXX is in design mode. After you have completed the application, you may run it directly, without leaving the development environment, in either a run mode or a debug mode. Debug mode runs your program under an integrated, easy-to-use debugger. Once you are satisfied with the application, you can generate an executable file for distribution.

An application in VX-REXX is known as a project. To build the user interface, you place objects on the windows. An object consists of properties and methods. An object's properties include data and style settings. Methods are the functions the object can perform. For example, an entry field object has properties such as its value, maximum number of characters, color, and whether or not the field is readable. It has methods for setting the focus, getting and setting properties, and invoking help. To set the properties, you open the object's Properties notebook and set as desired. You can also set properties at run time by calling methods to set the properties dynamically. You can even create the objects at run time rather than at design time.



Like Presentation Manager, a VX-REXX application is event driven. Each object has a set of events it can respond to, such as a mouse click, entering data in the object, or another object being dropped on it. Most of the `WM_CONTROL` notification messages familiar to the Presentation Manager programmer show up as events. VX-REXX adds more events, including ones for drag-and-drop support and data verification. You define how the object is to respond to each event by coding an event handling routine. Select an event from a cascaded pop-up menu or from the properties notebook, and a section editor displays an edit window for your program to handle the event. The programming for

handling an event is often quite short. REXX is an easy language to use, and VX-REXX takes care of the details. Coding event handling routines is much easier in VX-REXX than in C.

Multithreading is the key to OS/2 applications that are responsive to the user. VX-REXX makes multithreading easy—one command launches a thread. The thread resides in a separate code file and can post results back to the launcher.

Figure 1 shows the event list for the Cancel push button on a simple window. Clicking the right mouse button on any object pops up a context menu. The Events menu item cascades to a list of all events supported for that control. A



Figure 1. Event list for a push button.



check mark is placed beside events for which you have defined an event processing routine.

Figure 2 shows part of the event routine for a Change event on an entry field. I wanted to disable the OK button if nothing had been entered in the entry field. Drag-and-drop programming makes this easy. If you drag an object to the edit window, a list of all methods and functions for that object is shown. Select the function to enable or disable an object and a dialogue appears allowing you to specify whether the object is to be enabled or disabled. The correct code is inserted in the edit window where you dropped the object.

If you forget some REXX syn-

tax or a VX-REXX function, just click on the right mouse button in the edit window, select "insert code...", and a list of VX-REXX functions as well as those of the REXX language itself is displayed. Select the one you want and the editor will insert the accurate and complete REXX statement. It's hard to go wrong!

The drag-and-drop programming and code insertion list greatly simplify learning the product. The implementation is very smooth and allows you to use as much or as little of it as you like. You can rely on it for everything, for nothing, or for anything in between, depending on how adept you are with VX-REXX.

Version 2.1 fills in many gaps found in earlier versions. Here are a few samples from a list too long to include in this review. In version 2.1, you can drag an object from the toolbox to the window to create a new object of that type. Only one step (drag) is required rather than the two steps (select tool, create object) required in earlier versions. A Tab Editor eases changing the tab order of objects in a window. Properties have been added to many objects to support most Presentation Manager control styles. Many new methods and properties have been added to the Container and MLE objects. Drag-and-drop events are supported on all objects, making it easy to sup-

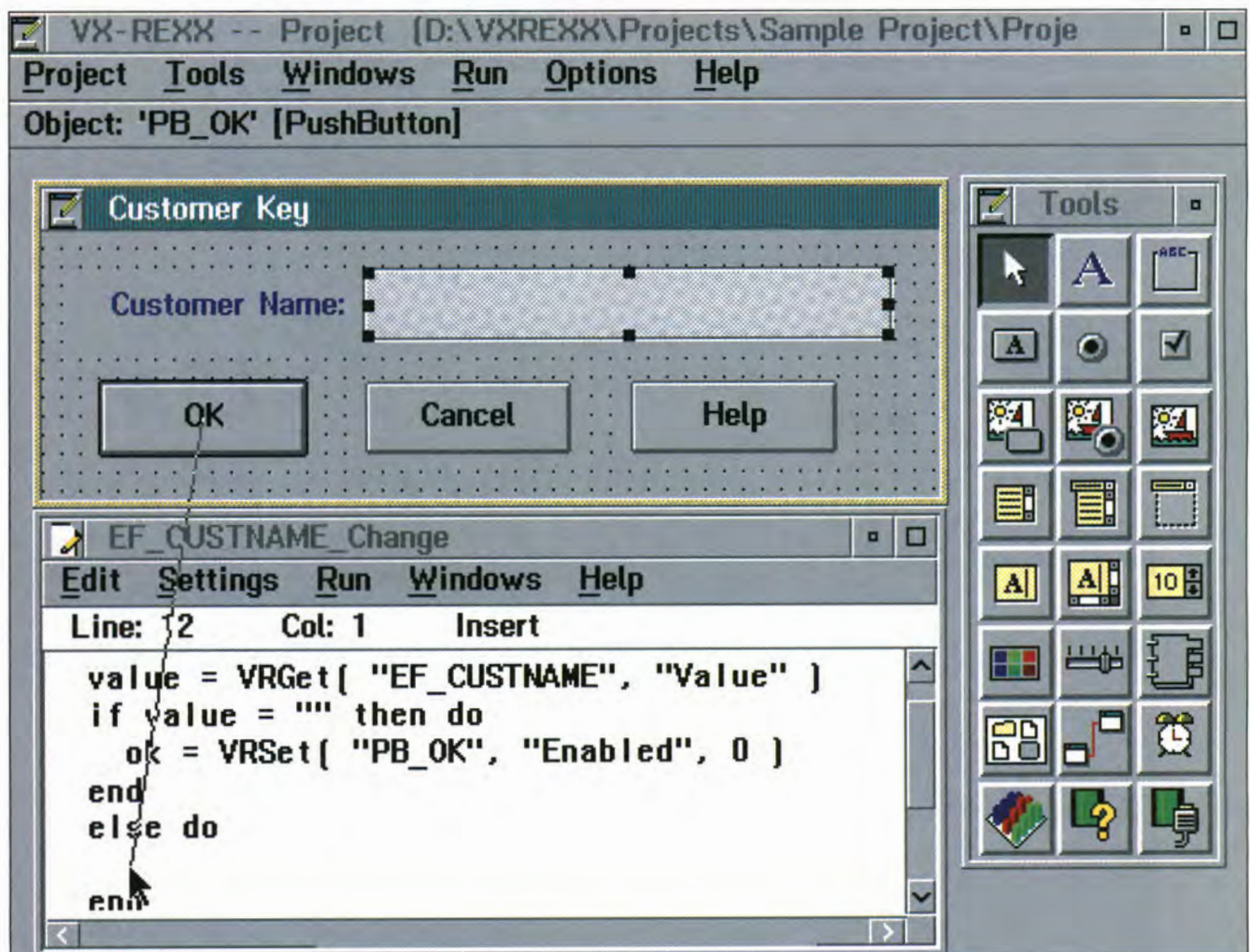
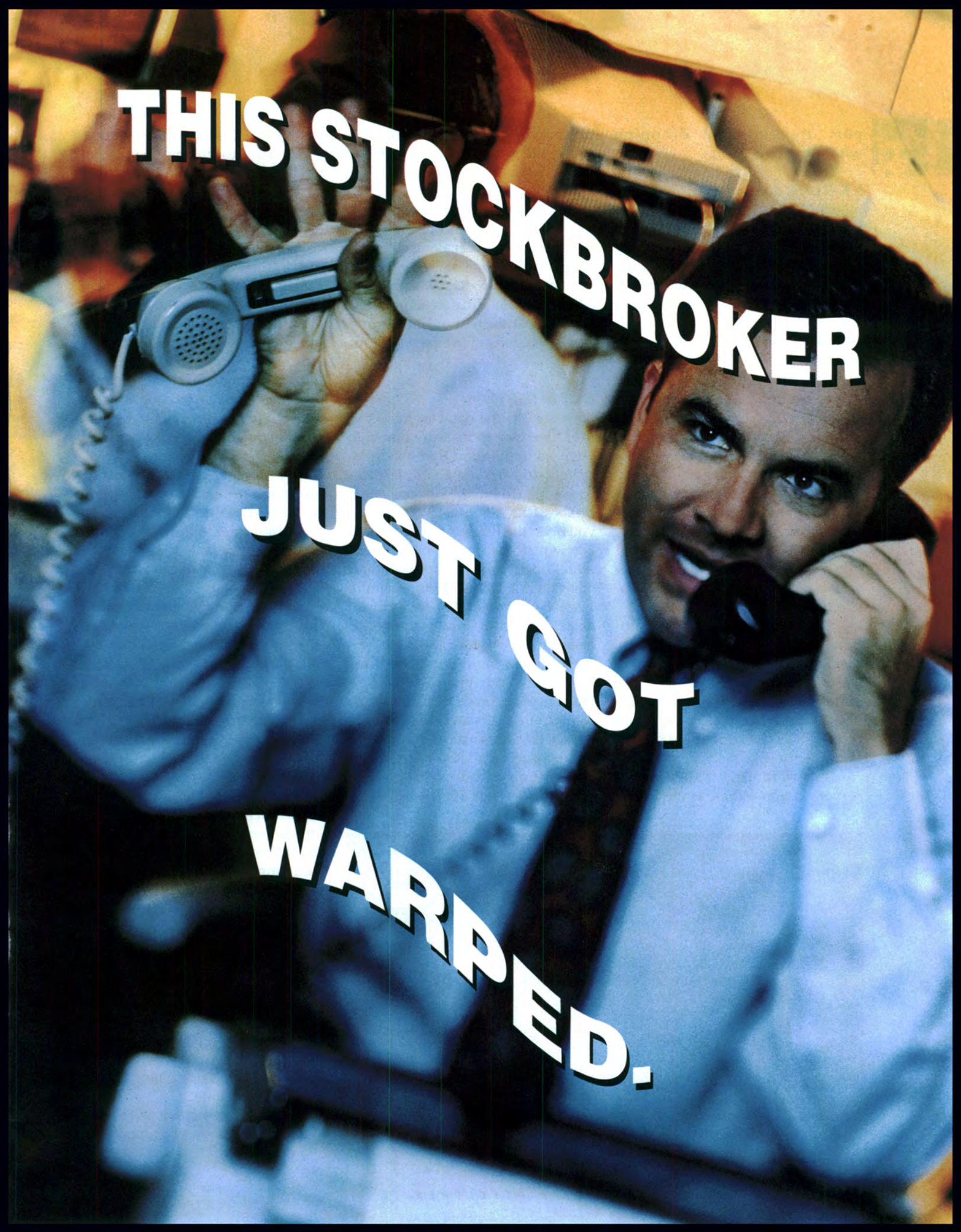


Figure 2. Enabling a push button using drag-and-drop programming.



THIS STOCKBROKER

JUST GOT

WARRPED.



port dragging and dropping within an application.

If you need to handle events VX-REXX does not support, the new `DefineEvent` method allows you to define a processing event for any Presentation Manager message. This is useful for handling unforeseen conditions or custom objects with their own private messages.

VX-REXX FOR OS/2 CLIENT/SERVER EDITION

The VX-REXX for OS/2 Client/Server Edition adds database and charting tools to VX-REXX version 2.1. The database tools include two new objects—a connection object and a query object—and a Database Administrator utility. The charting tool is a new chart object that supports a wide range of chart types. Complete online and printed manuals and sample programs demonstrate the varied capabilities of these objects.

The Client/Server Edition supports IBM Database 2/2 (DB2/2), IBM DB2 on larger systems using DDCS/2, Watcom SQL for OS/2, and any ODBC-compliant database. For ODBC, you must obtain the ODBC driver for the database from InterSolv.

But, wait! This is the Client/Server Edition. Where is the Server

part of the product? Despite the name, there is neither a VX-REXX server component nor support for building a VX-REXX server. VX-REXX for OS/2 Client/Server edition is a tool to build client applications. Any server component is supplied by the database system.

I tested VX-REXX for OS/2 Client/Server Edition using IBM DB2/2 and both stand-alone and server versions of Watcom SQL for OS/2.

EASE OF USE

How easy is the database object to use? One way to measure that is to build an application that will view, search, and update a simple database table. To build that application in VX-REXX Client Server Edition, you open the Projects folder, tear off a new project from the project template, open your new project, and open the VX-REXX object within it. VX-REXX presents you with an empty window. Drag a connection object to the window. Open the connection object and enter the database name and type (IBM DB2/2, Watcom SQL, or ODBC). Close the connection object and drag a query object to the window. Open the query object and select the table and columns desired. Check the check boxes that ask VX-REXX to

create the fields and push buttons. Check a radio button to indicate whether you want an entry field per column or prefer that multiple records be displayed in a container. Close the query object, and VX-REXX creates all of the fields and push buttons needed.

You now have a working application, and it has taken only a few minutes. A few more minutes takes care of cosmetics, such as adding a title bar, changing field captions from database field names to user terms, and refining the layout by adjusting entry field lengths and rearranging the push buttons. Figure 3 shows the result.

The sample programs include more complex examples, including two with master-detail organization.

Simple methods to insert and delete records make database programming easy. If you want to use SQL directly, just ask the connection object to execute the SQL statement. The connection object provides error codes and messages from the SQL statement. Converting from DB2/2 to Watcom SQL is easy with the connection object. All you have to do is change the database type and name properties. That can be done either at design time or during execution.

The new chart object packs a lot of functionality into one object. You can create the basic bar, area, pie, ribbon, and line charts. If you need something different, try the gantt, stock, box whisker, radar, 100% area, XY, proportional, bubble, or combination charts. Some of the charts can be displayed three dimensionally. More than 150 chart options give you detailed control over the appearance of the chart. A Chart Editor lets you control these options at design time. As with other VX-REXX objects, you can also create charts or change these options at run time.

The chart object has events for clicking and double clicking. When

Figure 3. Simple database table query and update application.



On a typical
day, stockbroker Bret

Williams
makes his
personal

computer do some very atypical things.

He imports live market data off the ticker at

the same
time that he's
reading the
stock page

on-line, at the same time that his computer is
firing off e-mail to a client, and at the same time
that it's faxing his lunch order (pizza, usually).

He runs his PC without fear of crashing.
(If any one of his applications ever goes down,
everything else stays up.)

And he gets more out of the software he
knows and uses, like Windows™ and DOS.

Bret has OS/2 Warp, the operating system
that offers true multitasking, Crash Protection,™
Internet-access – not to mention a BonusPak
filled with productivity applications.

The price is also less than you ever thought

possible:
under \$90.
Of course,
you're

welcome to do your own risk assessment.

OS/2 Warp is available now. (For stock-
brokers and anyone else looking for a great
investment.) To get warped, stop by your local
software dealer, or call 1 800 3 IBM-OS2.

Ask for a free demo disk.

The new 32-bit, multitasking, multimedia, Internet-accessed, crash-protected,
Windows-friendly, totally cool way to run your computer. **OS/2® WARP**

OS/2 Warp is available from your software dealer. It is also available from IBM for \$89 by calling 1 800 3 IBM-OS2.
Reseller prices may vary. OS/2 Warp consists of OS/2 Version 3 and BonusPak. IBM, Operating System/2 and OS/2 are registered trademarks of the
International Business Machines Corporation. Crash Protection and the OS/2 logo are trademarks of IBM. Windows is a trademark of Microsoft Corporation.
© 1994 IBM Corp.

Circle Reader Service Number 6

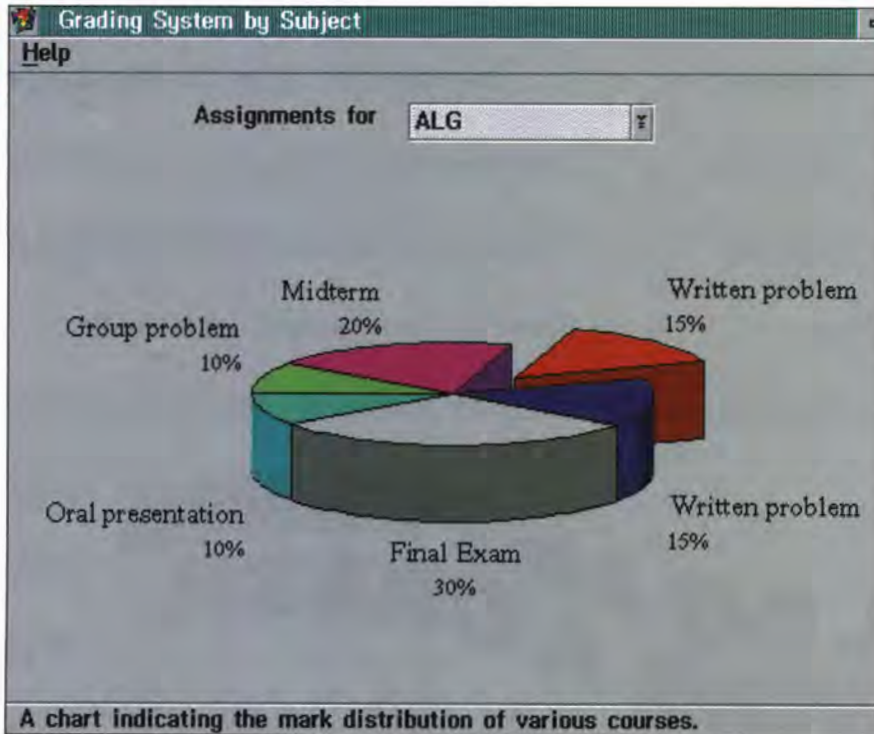


Figure 4. Pie chart in three dimensions.

you receive such a mouse event, you can ask which row and column was clicked on. This makes it easy to implement a "drill-down" chart, where clicking on one part of the chart will explode the chart into more detail. The chart object can also be bound to a query object. This gives you a chart display of data from a database with almost no programming required.

Eleven sample chart programs are supplied, including two in which the chart is bound to a query object and one that demonstrates a drill-down application. Figure 4 shows a three-dimensional pie chart that is typical of the charts you can produce with the chart object.

DOCUMENTATION

The Client/Server edition includes two books: the *Programmer's Guide and Reference* and the *Client/Server Edition Objects Guide*. The first book is common to both VX-REXX 2.1 and the Client/Server edition. The *Programmer's Guide* includes a

tutorial on creating a simple application. It describes the structure of a VX-REXX project and has sections addressing common programming topics, such as using multiple threads, modeless windows, debugging, controlling, other processes, and creating objects at run time.

The reference section lacks the detailed descriptions found in the version 2.0 manuals. With this new version, Watcom has reduced a 450-page reference to a 25-page alphabetical list of objects, properties, events, methods, predefined routines, and functions. While everything in the books is available online in an .INF file, I find it impossible to really learn a tool from online documentation. To become expert, I find I have to read reference manuals from cover to cover until I understand it all. You won't want to discard your old VX-REXX 2.0 manual!

The *Client/Server Edition Objects Guide* contains two sections describing the features of the VX-

REXX Client/Server Edition beyond those included in the base VX-REXX version 2.1: the Database Objects Guide and the Chart Object Guide. Each guide has tutorial, programmer's guide, and reference sections. In this book, the reference sections include complete descriptions and not just lists.

All of the books are available online as .INF files that use hyperlinks well. The information is organized in additional ways to the printed presentation. For example, the online reference shows the events, properties, and methods available for each type of object. The online documents are a pleasure to use.

WARTS AND BLEMISHES

The base components of VX-REXX 2.1 have been honed over several releases and operate very smoothly. The database tools of the Client/Server Edition are new and do have rough edges, primarily in the design environment. I was able to work around problem areas easily though. The Database Administrator tool is another story. It had serious enough flaws that I was unable to use it to define a database. Fortunately, this tool is a nicety and not essential.

Watcom released a version 2.1a patch after this review was completed. This patch corrects problems in version 2.1 and problems related to OS/2 Warp 3. I checked the problems I had found, and they were corrected. The 2.1a patch is available in the Watcom forum on CompuServe.

LIMITS OF APPLICABILITY

While VX-REXX does provide a Presentation Manager printer selection dialogue to make selecting the printer easy, Presentation Manager printing is not provided. To print, you write a report to a file, including printer-specific control codes if desired, then print the



file using a `VRPrintFile` function. The chart object includes a method to print a chart, but VX-REXX does not provide a way to combine a chart with other printed output.

VX-REXX provides an unadorned entry field. More complex entry fields for dates, zip codes, and other formats are available from third-party vendors as add-on components. Watcom also has an Object Development Kit available so you can develop your own custom objects to use in VX-REXX projects. VX-REXX does not support drawing directly on a window; you would need to develop your own object to access a window directly. Also, VX-REXX does not support every last feature of OS/2 Presentation Manager, such as the owner draw style. The ease of use and rapid development more than made up for any small features that were missing in my use. In writing a small business application, VX-REXX supported all of the OS/2 Presentation Manager features I needed and many more besides.

REXX is not C, and with this distinction come both advantages and disadvantages. It may be difficult to work with some data types and structures, as REXX has only strings and integers. Using C and Presentation Manager, you can create subclasses and superclasses to alter the behavior of objects. This will be harder to do in REXX. An interpreted language such as REXX is likely to be slower than C in some situations, such as loading a container with thousands of records.

Applications developed in VX-REXX can be distributed as EXE files. Some Watcom DLL files are

required; they may be distributed without royalty payment. The `VROBJ.DLL` file has grown to 893K in this release; database and charting DLLs and related files are about 200K and 350K, respectively, depending on the database to be supported.

TECHNICAL SUPPORT

Watcom offers technical support by telephone, fax, BBS, Internet, and CompuServe (GO WATCOM). A fax response system provides technical information. I called Watcom twice and found the staff to be courteous and knowledgeable. I was quickly passed on to a specialist for a more complex question. I posted several questions on the Watcom forum on CompuServe. How-to questions were generally answered overnight; more difficult questions requiring research took longer. The Watcom staff was very receptive to problem reports and very honest about problems and workarounds. I

found this response refreshing; it gave me confidence that I would get real answers to my questions.

SUMMARY

VX-REXX is excellent for developing OS/2 Presentation Manager applications quickly. The drag-and-drop programming interface makes learning VX-REXX and REXX easy. All OS/2 controls are supported, and custom controls can be added, allowing the developer great flexibility in designing the application. The Client/Server Edition makes using an SQL database easy. I will definitely keep the VX-REXX Client/Server Edition in my toolbox.

Guy Scharf is president of *Software Architects Inc.*, a software development firm specializing in developing OS/2 Presentation Manager applications for vendors and business. He can be reached via e-mail at 76702.557@compuserve.com or at Software Architects Inc., 2163 Jardin Drive, Mountain View, Calif. 94040.

PRODUCT INFORMATION

Watcom VX-REXX for OS/2 Client/Server Edition\$299
Watcom VX-REXX for OS/2 Version 2.1 Standard Edition\$99

Upgrade pricing:

From earlier version to 2.1\$59
From earlier version to Client/Server Edition\$199

Vendor: Watcom International Corp. (subsidiary of Powersoft Corp.)
415 Phillip St.
Waterloo, Ont. N2L 3X2
Tel. (519) 886-3700
Fax. (519) 747-4971



Keeping with the theme of the issue, a mystical area of programming, the exception handler, is demystified. **By MARK BENGE and MATT SMITH**

An Exception You Can Handle



Mark Benge



Matt Smith

Not so long ago, in a place nearby (hey, *Star Wars* is in production again, just getting ready, sort of), a demo was being waged. Wonderful screens of dialogues and windows were being dutifully displayed to wonderment of all. Everyone was being mesmerized by this show of power and majesty. Splat! Wait a minute, that wasn't part of the script.

While looking for a quick exit from this major embarrassment, it was noticed that the audience was moving in closer, trying to get a look at the display. They were doing this while saying to each other that they had to know how it was done.

Hold it, did they just ask how it went splat? Nope. They wanted to know how the exception handler was done such that the exception dialogue reported the function where the application tripped over itself.

GRACEFULLY EXITING STAGE LEFT

Interesting, that. Having a major faux pas and being able to walk away virtually unscathed. Why? Quite simply by having a graceful exception handler that told the user what was happening and providing us, the writers of the application, with a nice report of where things went wrong.

Unfortunately, almost all the major compiler writers just don't get it that OS/2 has a really neat exception handler that with a little thought could produce useful diagnostic information and should be part of the compiler they sell.

Now, you may be asking, why would we want to talk about it and not keep this as one of the trade secrets or crown jewels? We want to challenge those compiler writers into providing support for this issue. We will show you why in a moment.

YUCKO, THEORY TIME

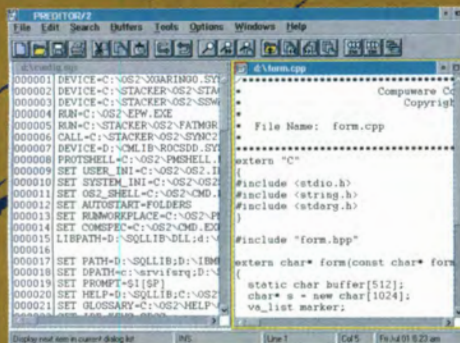
Okay, kids, put on your theory propeller caps and follow the discussion on the chalkboard. The problem: OS/2 catches our program doing something stupid and tells our exception handler where it occurred. Great, let's just print out the address and call it a day!

Yeah, right. That's exactly what most of the compiler exception handlers do if they handle the exception at all. We still don't know where the program shot itself in the foot though.

Well, maybe we should use Method I: the good ol' linker map, which is the method that we have all used for years and years. Unfortunately, it really is only useful in telling us sort of, maybe, kind of, somewhat where the problem

For the untamed
programmer
only.

PREDITOR/2



Introducing PREDITOR/2.

Your source code has never fallen into the grips of a more powerful editor. PREDITOR/2 is for developers who don't have the time to deal with rigid programming environments, who must change facilities to fit their exact requirements and who need to dramatically extend editing functions to attain peak productivity. With this editor, you'll work with the speed and strength of a true predator.

HIGHLIGHTS:

- CUA-compliant GUI
- Multiple document interface
- BRIEF, VI, EMACS, ISPF, CUA emulation
- WorkFrame/2 integration
- Built-in compiler support
- Extensive customization facilities
- Powerful C-like extension language
- Unlimited Undo and Redo
- Powerful search and replace functions
- No limits on file sizes, buffers or windows

Special Introductory Price... \$149.00 • MSRP \$249.00

To order PREDITOR/2 or to receive a brochure, call now. 1-800-535-8707 or fax: 1-810-737-7119

PREDITOR/2 is a trademark of Compuware Corporation. All other company or product names are trademarks of their respective owners.
© 1994 Compuware Corporation.

Circle Reader Service Number 35


COMPUWARE
Sophisticated Software Used Every Day





occurred. This is due to the fact that the link map shows only the addresses for the global functions and not for any statics. It is also difficult to try to locate the problem if the address is outside the main application address space, such as within a system call or within a DLL you have called.

The method we will show, Method II, borrows heavily from the link map concept but without paperwork. Here we rely on tables of tables. Huh??

Okay, the concept is really quite simple; it is just the implementation that gets a wee bit messy and tedious. Consider that the exception address is within your main application space. By main application space, we mean your source from the first module up to the last module, as illustrated in Figure 1. Following this

last module usually is the start of the compiler run-time library. By rights, it is part of your address space, but since we don't know its address or function address, we treat it as being outside our address space like that of the system.

Each of your functions follow one another, maybe in the order you coded or in some other order the compiler and linker have put together. The only thing we need to do is find out between which two functions the exception occurred. Then we know where we tripped up. That's it. It is as simple as that. The idea even works with your DLLs.

Method III is where the compiler writers are needed and should take note. Using Method II, where the module and function are reported, you should also have the source line reported. Now that would be truly wonderful.

ACTUAL DESIGN

The real trick to making most of this work is the building of a table within each source module that records the function address along with the name of the function. Then in the exception handler of the application, a different table is built that references each of the tables within each of the different modules. Like we said, tables of tables.

When an exception occurs that is handled by our exception handler, we sort the tables so they are in ascending order and then walk them, trying to find the two functions in between which the exception falls. Once found, we know exactly where we died. This works great, especially when you have your software being used halfway around the world and it can tell you exactly where it is feeling sick. Only if it were truly this simple!

What do you do when there is the possibility of the exception occurring within a DLL? Well, first you check to see if the exception falls within your source code address space, namely from the start of the application to the exception handler, which we have caused to be at the end. If it isn't in this space, we call the exception lookup handlers within each of our DLLs to see if the exception occurred in one of them.

If none of them reports owning the naughty code, we then use our next death-defying act (at times, as we will show in a moment, this is exactly how we have coined it). Here we walk the stack with the idea that we will trace back through to our address space. Then we state that the function was the starting point of the stack trace. At least this gives us an idea of where to start looking.

And, if all else fails, like when the stack is a mixture of 32-bit and 16-bit addresses, we stick our heads in the sand just like the compiler writers do.

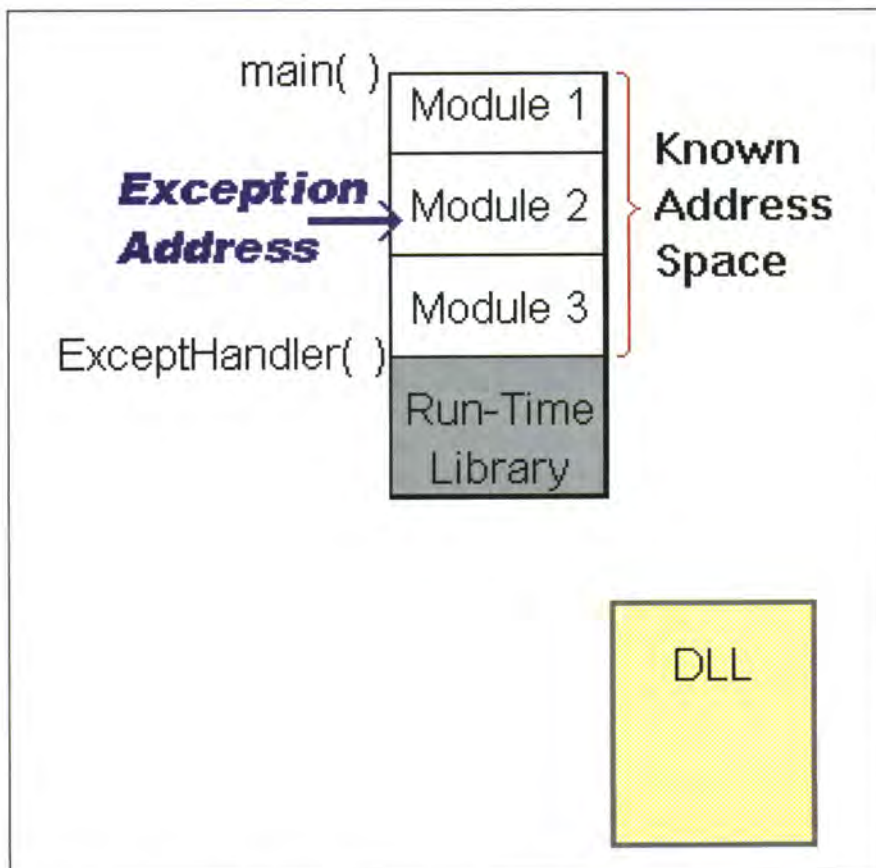


Figure 1. Main application space.

THE IMPLEMENTATION

In the sample source code, which can be found lurking in the usual electronic hangouts, we have a bunch of modules with some stupid, do-nothing functions that, depending on the alignment of the planets, will decide to GP.

Within each is a table that is declared statically. This table is then referenced by a structure, `INTERNALADDRESS` that contains the module name and the table address.

Then, within the `Xcept.C` module is an array of the `INTERNALADDRESS` structures from each of the modules. This is the backbone of our lookup scheme.

Inside our exception handler, we place the smarts to sort the tables and call a routine that will do the actual lookup. If the address isn't found within the lookup tables or any DLL's, we use the returned BP register to start the walk back through the stack.

Now, this walk on the wild

side of the stack can be tricky. First, you have to be careful that the value you see is a 32-bit value and not a 16-bit return address.

This is done by checking the address against the stack limit. If it is outside it, it is probably a 16-bit address. Next, and there is no check on this, if the compiler weenies have decided to party on the BP register by using it as a general-purpose register, you are most definitely out of luck. Carefully check your compiler documentation since some of these compilers don't explain very well in their documentation what happens when they use the BP register for anything but the stack frame.

Once we have traced back into our address space through the stack, we are essentially done. We just have to look up the traced back address within our tables.

Now, it is a matter of reporting the damage. Here we do it in two places: a log file that we append

each time an exception occurs and the exception dialogue shown in Figure 2.

A LATE CHRISTMAS PRESENT

One of the really neat things about OS/2 is that if you provide the proper code, you can recover gracefully from a full-stop exception and continue using your application. A good example would be a conversion filter. Here, you would design it so any memory being used is not intermixed with the main application. Therefore, if things decide to fall apart, your main data is still in one piece.

The best way to do this is to place the filter within a dynamically loaded DLL. Just as soon as the function in the DLL is called, you register an exception handler within the DLL. This exception handler is now at the top of the chain, which means it gets called first. Next, you do a `setjmp` call, so if an exception occurs, you can

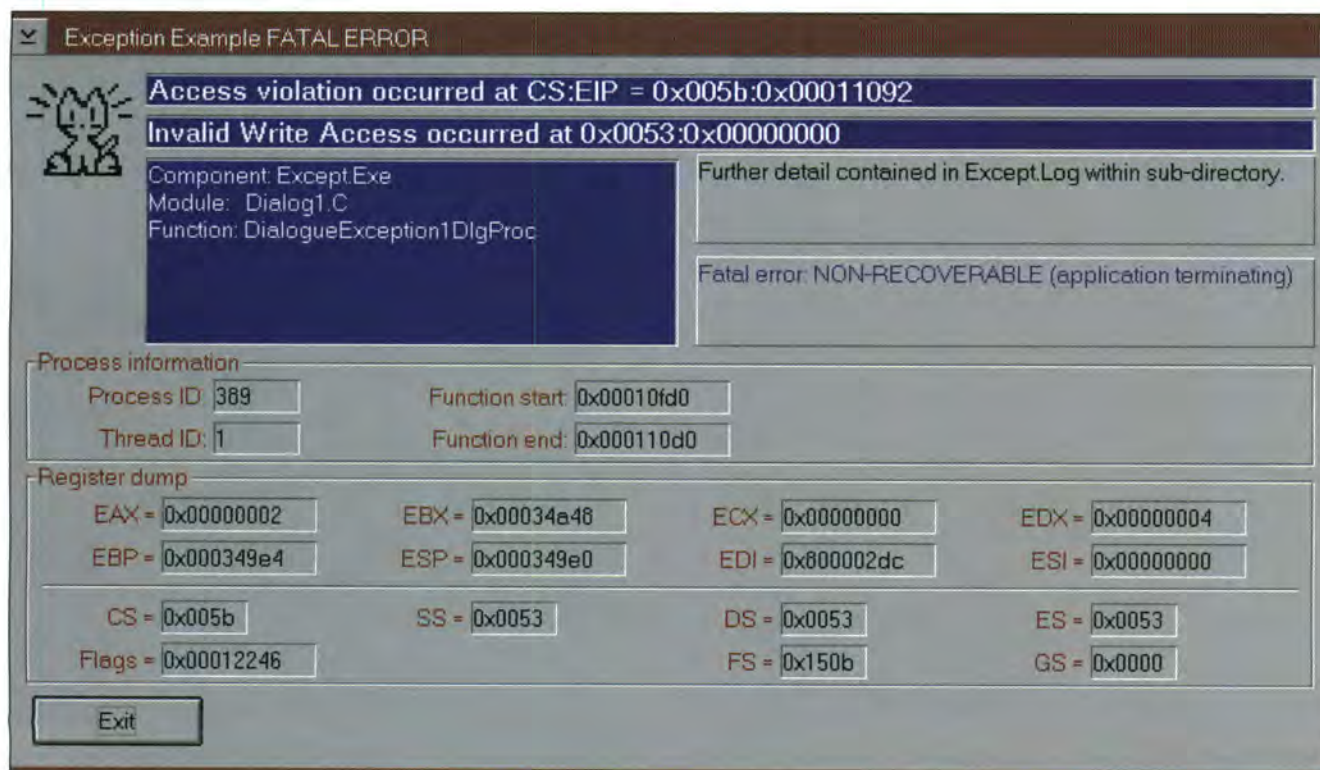


Figure 2. Exception dialogue.



gracefully recover and report it to the user.

The DLL exception handler in this case can be like the full-blown one. Instead of stating it is fatal, however, state it is recoverable. Then perform a `long jmp` back to the location where you did the `set jmp` and unwind the DLL exception handler before returning to the main application.

We want to caution you here. You have to be careful with this and have clearly thought through your design of both the DLL and your data. You should really only use this if you can separate your application data from that of the DLL.

MAKING IT ALL WORK

To make this work with a much larger application, you must update the tables by hand each time you add or remove a function. This updating of the tables is why we mentioned earlier the tedious nature of the exception handler. Hence our endearment with the compiler writers. The compilers can easily and dynamically update these tables for us. They will ensure that they have

all of the functions defined, whereas we may make a mistake if we are not diligent.

The same problem occurs when we add a new module. We need to create a new entry within the internal address array after we have created the table within the new module.

But this work will be worth it when you get your first support call from some pub in Edinburgh and you know exactly where to look within your application.

Mark Benge, IBM Software Solutions, Cary, N.C., is a staff programmer who joined IBM in 1989 and has worked on various CUA '91 controls for OS/2 2.x. He works in IBM user interface class library development, where he was involved in the implementation of C++ classes for drag and drop in C Set++ 2.1. Mark has a B.S. in computer science from Western Carolina University. He can be reached on CompuServe at 73532,2063 and on Internet at banzai@vnet.ibm.com.

Matt Smith, Prominare Inc., Toronto, Ont., is lead architect for the Prominare Development System, an OS/2 2.x

advanced GUI development environment. Matt has been actively involved with OS/2 since 1988 and cofounded Prominare in 1990. Smith has a degree in architecture from the University of Waterloo. He can be reached on Internet at msmith@interlog.com.

REFERENCES

Credits: Screen captures were done through OpenShutter from One UP Corp. and touched up through PaintBrush in Win-OS/2.

You can download XCEPTN.EXE from these electronic sources:

CompuServe's OS2DF2 forum (OS/2 Developer Mag section).

File Area 11 on the IBM PCC BBS, (919) 517-0001.

OS/2 Tools section on the OS/2 BBS for IBM TALKLink customers.

FTP to the site address gold.interlog.com and log in as anonymous. The source is located in the /pub/prominare subdirectory.

WHEN I WANT ADVICE ABOUT OBJECT-ORIENTED
PROGRAMMING, I'LL ASK AN EXPERT.



SO, ASK US.

You already know reusable class libraries promise a dramatic increase in productivity and efficiency for professional application developers. You also know class libraries created with one C++ compiler can't link with code created by another C++ compiler. And that libraries written in one language can't be used with client code written in another language.

But here's something you might not know: MetaWare and IBM are changing all that.

Out With The Old. In With The New. The tight binary coupling that exists between object-oriented class libraries and client code means that changes to either—no matter how small—require a complete recompilation not only of the class itself, but of all the client modules as well.

But with IBM's System Object Model (SOM) and MetaWare's High C/C++ DirectToSOM compiler for OS/2, developers can break that tight binary coupling and extend the advantages of procedure libraries to object-oriented technology.

Pretty amazing, right? And right now, you think this is where our pitch to sell you a compiler comes in. But you're wrong.

Information Only. And It's Free. We know what you really want is information. That way you can make up your own mind about the benefits of SOM and the DirectToSOM High C/C++ compiler for OS/2. We've prepared a white paper that will help. It's called Truly Reusable Objects Directly From C++: How to Create and Distribute Them. It's yours just for the asking.

To receive your free information, call, fax, or e-mail MetaWare at the numbers shown below. Once you review the facts for yourself, you'll understand why MetaWare and IBM mean "objects made easy."



CALL 408 429 6382

FAX 408 429 9273

OR

E-MAIL TECHSALES@METAWARE.COM

FOR YOUR FREE COPY.

TODAY.

 **MetaWare**
OBJECTS MADE EASY.

Circle Reader Service Number 5



Often some of the simplest ideas yield the greatest benefits. This issue's *Programming Insider* shows you how aliasing functions can give your applications more speed in less space. **By DAVID REICH**

An API by Any Other Name...



David Reich

This issue, I want to discuss an important but usually unknown or overlooked aspect of application performance tuning: the subject of calling APIs in DLLs, whether they be your own DLLs or system service DLLs. Depending on your intended use of functions that reside in DLLs, you have many options in how to call and use them. There are also obvious performance impacts if you call an API by name, having it preloaded and fixed up when you call the function, or if you load the module when you need it and fix up the reference at run time. There are also impacts in calling a single function, or sets of functions, all over your code. Let's first look at how DLLs work.

LINKING DYNAMICALLY

Like static run-time libraries, a DLL is a library of functions that can be used by many programs. However, the functions in DLLs are not bound to executable programs. The functions remain in the DLLs, which are loaded by the OS/2 program loader and are callable by programs written to access their functions.

A statically linked library is a .LIB file, bound to an executable. A DLL consists of the DLL, which contains the routines, and a .LIB file, which is the "road map" to the DLL. When you build an

application and link with a DLL's .LIB file, the linker places not the routine (as it would with a statically linked library) but an external reference record that points to the DLL and a function number (the ordinal of the function) within the DLL into the object code.

If you were to peek inside your executable program file, you would see that in the CALL instructions for all of the DLL reference records, there is just a reference to an external function. More importantly, if you were to look at the EXE header (you can, with the tool EXEHDR, which is part of the compiler and toolkit), you would see there are references to DLLs and ordinals, such as `PMWIN.139`. This indicates that the function at ordinal 139 (the 139th function) in `PMWIN.DLL` is the one being referenced.

This list of DLL ordinals or imports (referencing a DLL's function by name is also called importing the function) in the EXE header is called the import list. Whenever an OS/2 program is loaded, the program loader examines the import list and fixes up the references to the DLLs. This causes the DLL to be loaded for the reference to be fixed up. This is not a terrible thing when calling system services since most of the system DLLs are already loaded. Also, the fix-up is much faster than if you were using



You don't need weird hair to form a powerful peer group

Announcing LANtastic® for OS/2®

Guilty of going to extremes to win peer-to-peer connectivity for your team? Then consider LANtastic for OS/2, a true 32-bit multitasking, multithreaded peer-to-peer network operating system utilizing OS/2's superior power and performance.

LANtastic for OS/2 coexists with Novell® NetWare Requester™ for OS/2 and IBM® LAN Server clients and easily

communicates with LAN Server, Windows NT™ and other SMB-based servers. It has centralized network management and flexible security options that are easy to use.

Still need more proof? Call 1-800-809-1239 for the Artisoft Premier™ or Artisoft Advantage™ Partner nearest you. Then judge for yourself.



ARTISOFT®

© 1994 ARTISOFT, INC. All rights reserved. Artisoft and LANtastic are registered trademarks, and Artisoft Advantage and Artisoft Premier are service marks of Artisoft, Inc. OS/2 and IBM are registered trademarks of International Business Machines Corporation. Novell is a registered trademark and NetWare Requester is a trademark of Novell, Inc. Windows NT is a registered trademark of Microsoft Corporation.

Circle Reader Service Number 8



your own DLL since the loader would have to load and then resolve these addresses. This description applies when you import a function by name, such as calling `WinCreateWindow` or `DosSleep`.

It is also possible to call functions in DLLs by address. After all, DLLs are shared code. Your program can gain addressability to the code in the DLL through importing by name or by calling `DosLoadModule`. Once you have a handle to a DLL module, you have addressability to the module. Simply stated, addressability means that you can call a function in its address space without taking protection exceptions.

When you call `DosLoadModule`, you are given a module handle. The other part to calling a DLL function is to get the address of the function you want. You do this by calling `DosQueryProcAddr`. This function takes the DLL handle, the module name or ordinal, and gives you the 32-bit address of the function you want. You can then call the function, as shown in Figure 1.

You'll notice that the function call does not involve importing the function by name, and as such, there is no entry for the function in the EXE header. You may wonder where I'm heading with all this.

FIXING UP AND ALIASES

As you've just seen, whenever you call a function in a DLL by name, you import the name and it stays in the imports table in the EXE header. For every reference in the code to one of these imports, there is a fix-up in the application process's swappable memory. Not only are fix-ups performed when the application loads, slowing application load performance, they are also stored in application-

swappable memory, increasing the working set size of the application.

You could eliminate the fix-ups in application-swappable memory by calling the functions by address, but that is a bit more inconvenient, both in the amount of code you have to write and in the inherent parameter checking you get when calling the APIs. (The compiler checks the code against the function prototypes to give you parameter checking, whereas when calling by function pointer, as in Figure 1, you get no parameter checking.) Aside from the inconvenience, you'll have `DosLoadModule` and `DosQueryProcAddr` calls in many places in the code where you would not otherwise, wiping out the memory savings you'd gain by calling by address.

You can eliminate much of this overhead and keep the advantage of compile-time parameter checking with a simple technique I call "aliasing" APIs. That is, just call the API by another name.

Let's look at the difference between a DLL function call, which we've been discussing, and a call to an internal function of the application. DLL function calls have external reference records in the code of your application. In the 16-bit world, these would have been far calls. In the 32-bit world, these are simply near calls since the entire system is one 32-bit flat address space. An internal function call is also a near call. The biggest difference, and the one in which we are interested for this discussion, is the overhead inside your application to store all of the fix-ups for external, or DLL, function calls. Internal function calls need no fix-ups. If you can call system services with internal function calls, you reduce the need for these fix-ups.

If you have many places in your application where you call, say, `DosBeep`, you will have a fix-up for every call. Why not just write one small function in your application, called `MyDosBeep`, which takes the same parameters as `DosBeep`? Inside `MyDosBeep`, you simply call the real system service `DosBeep` with the same parameters as `DosBeep` taken from the parameters passed to `MyDosBeep`. This may seem an inordinate amount of overhead to save a few bytes here and there, but depending on how often you use a system API, you can save a considerable amount of memory.

This simple technique not only saves you memory when you need to call system services but also can help you structure your own code to make it easier to maintain and enhance. For example, writing your own "wrapper" functions to encapsulate calls to the OS/2 memory management APIs allows you to track the memory function calls in your application better. Rather than have to trace OS/2 memory management API function calls all over your code, you have a central point to watch.

This aliasing of DLL functions is not restricted to system service DLLs, but it can be used for any DLL. By using this simple technique, you can lower the working set size of your application while making debugging easier. Sometimes the things that give you benefits like this are the simplest and most often overlooked.

David Reich has been with the IBM OS/2 development team since 1987. He has worked on many parts of the system, supported customers and application developers, and traveled the world giving seminars and teaching OS/2. He is currently working on the second edition of, *Designing OS/2 Applications*, published by John Wiley & Sons. He can be reached on CompuServe at 76711,632 or via Internet at speedracer@vnet.ibm.com.

```
rc = (*pfnFuncAddr)(parm1, parm2, parm3, parm4);
```

Figure 1. Calling a DLL function by address.

OS/2 Programming Shouldn't Have to Hurt.



Introducing VisPro/C and VisPro/C++,
easy-to-use tools for IBM C Set and C Set ++
that aren't hard on your wallet.

Other tools allow you to build robust, 32-bit OS/2 applications, but they also require more of your time and money. VisPro/C™ and VisPro/C++™ give you complete drag and drop visual programming, with all of the bells and whistles -- without having to reach deep into your pockets.

Bells and Whistles

VisPro/C and VisPro/C++ pack an unbelievable amount of functionality into a box. You get a Workplace Shell-enabled GUI environment that fully supports the IBM C Set compilers and User Interface Class Library. Both products have the most CUA '91 objects from simple buttons to 3-D business graphics and everything in between.

And if that isn't enough, we've included a visual DB2/2 database designer that allows you to create embedded SQL client/server applications or reverse-engineer existing DB2/2 databases.

From One World-Class Tool Comes Two More

VisPro/C and VisPro/C++ follow in the footsteps of VisPro/REXX, the pioneer visual REXX programming tool. Thousands of customers already rely on VisPro/REXX for powerful OS/2 development.

Now we provide the same renowned functionality for the C and C++ languages. At the same easy-on-your-wallet prices.

See it to believe it!

Buy now and take advantage of our no risk, 60-day money back guarantee.

VisPro/C or VisPro/C++ ...\$399

Development Suite\$599

All Three Products:

VisPro/REXX Gold, VisPro/C, VisPro/C++

Competitive Upgrade\$199

Upgrade any visual programming tool to VisPro/C or VisPro/C++



HockWare Incorporated
P.O. Box 336
Cary, NC 27512-0336
919-380-0616
919-380-0757 FAX
Go HockWare on CompuServe
hockware@vnet.net on Internet

HockWare, VisPro/C, VisPro/C++ and VisPro/REXX are trademarks of Hockware Incorporated. All other company, product and brand names are trademarks and/or registered trademarks of their respective holders and are mentioned for reference purposes only. © 1994 HockWare Incorporated. All rights reserved.



The use of advanced technology, such as IBM's OS/2 platform, has changed the methods and practices associated with testing applications. The introduction of robotic testing has answered many of the challenges associated with this new technology. **By JOSEPH P. BUSA**

Advanced Technology Needs Advanced Testing



Joseph P. Busa

As part of a continuous quality-improvement effort in corporate systems at Empire Blue Cross and Blue Shield, the introduction of advanced technology has caused a beneficial chain reaction in the project life cycle. Information systems is the nucleus that has to interface with all facets of our business. It is the hub of the wheel, with spokes that reach out into claims operations, membership services, and financial and other decision-making aspects of the corporation.

Introduction of new programming technology, such as the OS/2 platform, has created challenges for traditional testing methods. We can no longer expect to test these applications using methods of old. Whereas traditional mainframe systems may have a strictly defined path to follow to accomplish a task, graphical interface technology has exploded the possibilities and paths to take. A narrowly defined testing scenario may have been acceptable yesterday, but a wider scope is required today.

The realization that a wider scope has to be tested and that the personal computer platform is being used makes it clear that a new way of thinking has to take place. Customers are becoming more aware that they are in the driver's seat. They are expecting quality prod-

ucts so they can satisfy their customers' requirements.

THE FRONT LINE

Empire is the largest health insurance business in the country. We handle in the neighborhood of 100,000 claims and inquiries per day. The complexity of processing, the variety of products offered, and the numerous business relationships with health care professionals all come together in our data processing systems. With increasing competition in the health care market, our internal customers, claims examiners, and customer services representatives recognize and require many levels of independent testing of their products before they acceptance test it themselves.

The many features that OS/2 offers to the developer, such as drop-down menus, scroll bars, and profiling capability can make processing more efficient and easy to perform. These features create customer satisfaction. They give the application real value but cause sizable challenges to the testing community. Multitasking is a good example of a feature that truly creates value in an application but is rather complex for testers. The testing staff on the front line requires an understanding of these functionalities to test the application effectively.

TRANSITIONING TO SMALLTALK TECHNOLOGY?
INTRODUCING SMALLTALK TO YOUR ORGANIZATION?
Travel with the team that knows the way...

THE OBJECT PEOPLE

Your Smalltalk Experts



SMALLTALK

Education & Training

VisualAge
IBM Smalltalk
Smalltalk/V
VisualWorks
ENVY/Developer
Analysis & Design
Project Management
In-House & Open Courses

Project Related Services

Object Immersion Program
Project Mentoring
Custom Software Development
Tools Construction

The Object People Inc. 509-885 Meadowlands Dr., Ottawa, Ontario, K2C 3N2
Telephone: (613) 225-8812 FAX: (613) 225-5943

Smalltalk/V is a registered trademark of Digitalk, Inc.
VisualWorks is a trademark of ParcPlace Systems Inc.

VisualAge and IBM Smalltalk are registered trademarks of IBM.
ENVY is a registered trademark of OTI Inc.

Circle Reader Service Number 10

PLANNING AND TESTING METHODS

One of the most important aspects of our testing is planning. Our company is fortunate to have developed a project life cycle methodology, and it is used in the development of new systems, applications, and projects. This methodology calls for involvement of testing personnel in the requirements phase.

Test scenarios can be designed as requirements are stipulated. Testers get involved early in the development process, giving them more time and a better understanding of what needs to be tested. Well-trained testing analysts are critical to the success of testing. Organization of data, such as the use of a matrix format, is

one way to document the arrangement of criteria for testing. Each scenario that is created is a box on the matrix. In our case, we inventory the scenario so it may be used again depending on what type of testing we are planning to perform. This is especially useful for regression testing.

The execution of test scenarios in a complex environment is another critical aspect of testing that new, advanced technology has improved for us. Performing a field-by-field comparison of data, especially attempting to cover all the possible paths that can be taken in an OS/2 environment, has taken us weeks in the past. The introduction of robotic testing tools has answered the challenge of these new technological platforms and significantly reduced the effort. A test that took a week before is now accomplished in a day. Automated Testing can be used in multiple ways, and new automated testing strategies are being formulated by our test team on a regular basis. Vendors of this technology have been particularly helpful in assisting with the development of new approaches and in networking us with other customers for idea sharing.

OUR AUTOMATED TESTING FACILITY

After evaluating some of the industry's automated testing tools, such as Sterling's TestPro and Sequel Corp.'s ProTerm, we decided on Softbridge's ATF product since it happened to fit the needs of our operation. ATF is one of the few testing products we have found that can operate on multiple platforms, including OS/2. This facility has two major components: the executive and the agent (as illustrated in Figure 1). The executive directs the activity of test workstations or agents and contains the editing facility to develop the instructional script. The execu-

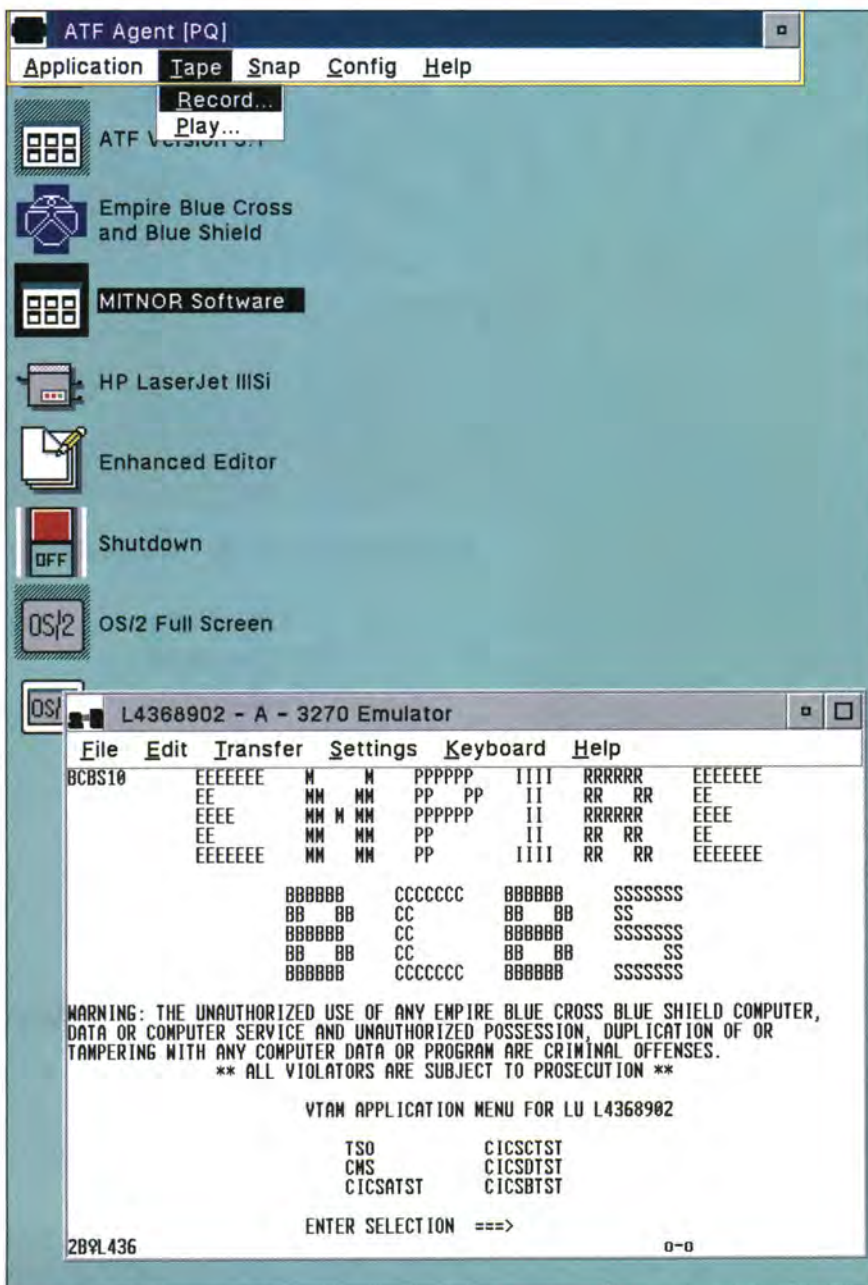


Figure 1. Executive station connected to multiple agent stations. Independent tests can be run on each agent.

tive acts as a command center from which testers can control and watch the execution of complex test scenarios. ATF has a sense of time and can also operate unattended. We plan on connecting as many as 50 workstation agents to one executive, giving us many options for testing. Throughput of data can be greatly enhanced by giving 20 stations 50 test scenarios each instead of using 1,000 scenarios at one station. A configuration of 50 agents can also be used to simulate production environments and to execute stress testing.

The core of the facility is the Distributed Test Language (DTL). This coding language allows users to perform any functionality they could normally do if they were sitting at the workstation themselves—and more. Mouse move-

ments, button selection, and keystrokes are all supported.

Users also have the ability to reboot the agent from the executive station. This can be convenient when certain error conditions are encountered during the execution of a script. Users can also use it to log the execution of script commands, allowing them to see results of the tests. One of the big problems with testing graphical user interface types of applications is reexecuting a test on a newer release of OS/2 or an application where the graphical windows have been changed ever so slightly—positional changes, for example. ATF has the ability to recognize graphical differences and, in most cases, permits scripts to run without change. Other features such as bitmap snaps and text snaps of the

screen, window, or area can be invaluable in the development and debugging process.



PRACTICAL APPLICATION

At Empire, one of the first testing procedures we applied this technology to was performance testing. ATF can accurately capture, to the second, response times of an application. With over 200 separate timings, this is impractical and nearly impossible for human testers.

We've established a procedure where each time a new release of software comes to the testing area, it is performance tested. That data is put to a database and then graphically displayed to show the ongoing history. For example, in our document imaging environment, performance tests on image retrieval times were developed and

Point and click your way to fast development of GUI client-server applications.

Do it with Guidelines, a second-generation object-oriented graphical workbench. Guidelines lets you create graphical client/server applications for OS/2 and Windows clients. A built-in prompter lets you simply point and click or drag and drop from the desktop using more than 30 presentation manager controls.

Guidelines utilizes JBA's high-level object-oriented language (JOT) to describe applications. After the application is designed, JOT automatically converts to C++ code, so you don't have to learn its complex syntax and rules. But if you already know C++ and want to use it directly, you can. The conversion takes place regardless of the platform intended.

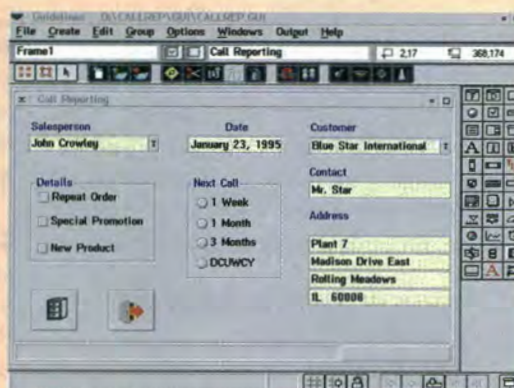
Guidelines saves time and simplifies the creation of graphical, platform-independent applications. Its scalable architecture lets you create anything from stand-

alone PC applications to complex enterprise-wide business solutions. Guidelines supports DB2/2, DB2/400, DB2/6000, DDCS, ODBC, Q+E Lib, and SQL.

Call for a free demo pack: 1-800-JBA-INTL.

In Canada: (905) 940-2442

JBA International
161 Gaither Drive, Ste. 200
Mt. Laurel, NJ 08054



© JBA International

JBA-111

Circle Reader Service Number 11



used to report on the performance of new software releases. As a regular process, retrieval performance continues to be monitored and has been improved with each major software release. Overall performance has increased by over 30% since the first measurement.

We have also developed a regression exercise that uses an internal data table feature. The internal data table is a spreadsheet-like facility from which scripts and tapes can read data and then apply that data to the application. In our regression tests, we have developed claims data that are run through and automatically compared with baselined results. Any expected result that is not matched with that of the baseline is documented in the log in a tester-customized report format. The results may then be easily examined by the test analysts. This has made our regression testing much more efficient. Attrition and downsizing has decreased testing staff by 15%, yet the testing being accomplished has

not diminished, and in some cases it has increased.

Another method we have developed is a "mapping" test. In our case, the OS/2 application, PC System 90, interfaces with a large IBM mainframe package system, as shown in Figure 2. PC System 90 acts as a friendly front end to the mainframe claims system and converts certain data fields. It then uploads the data interactively to the mainframe. We need to know from version to version if this process is occurring without defects. We have written a script to capture and compare the input to the PC System 90 front end with the resultant input on the mainframe. Once again, a baseline approach is used, and the data is automatically compared by ATF. Any errors are flagged and examined by the testing staff.

To fully leverage our quality-improvement initiative, we see opportunity in introducing this testing method in the software development phase to use in a unit test mode. This will uncover and prevent defects from continuing further into the project life cycle and will advance the concept of doing it right the first time. Coming from a software development background, I've experienced the lack of detail knowledge with the data that will pass through the system. Using a robotic tool with a testbed of data developed by team members that do have this detail knowledge will be a great advantage in unit testing.

VALUE OF IMPROVED QUALITY AND PRODUCTIVITY

The adoption of methods such as automated testing and full life cycle involvement has brought a sense of realism to our improvement initiative. Any organization that can do more with less without compromising quality will soon

come to the realization that working smarter not harder is what will benefit the bottom line and the front line. It is a win/win situation for all involved. In our case, regression testing, which in the past involved one resource for three days, now takes that same resource less than one day to perform with even more accuracy. The staff's morale and job satisfaction has increased. The improved process leads to lower costs and higher productivity. The resultant evidence is shown when quality software is put into production on a timely basis. This leads to customer satisfaction, and that is our ultimate goal.

CONCLUSION

As new technology is introduced to the business world, the organizations that take the risks and are determined to improve the current methods of operation will be the successful ones. Process improvement and reengineering can lead to big payoffs in quality and productivity. We need to take a look at our traditional processes, break them down into their components, and then apply visionary thinking to improve them. The development and use of products such as OS/2 and ATF were unthinkable 50 years ago. For all of us to fully participate in the improvement of our work, we need to look toward the future and continue to apply quality concepts.

Joseph P. Busa, CQA, is the quality assurance manager for Empire Blue Cross and Blue Shield in New York, the largest Blue Cross/Blue Shield in the country. He is responsible for all data processing quality assurance at Empire. He received his CQA (Certified Quality Analyst) from the Quality Assurance Institute in Orlando, Fla., in 1990. He has a B.S. in business administration from New York State University at Oswego, New York.

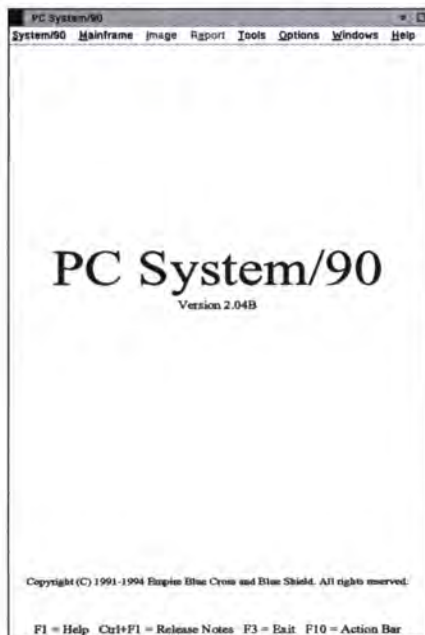


Figure 2. Empire's OS/2 platform with application PC/System 90 at the "Welcome Screen."

*This article demonstrates using OS/2 as the development base to build products that support multiple platforms. By **BRUCE LANDECK***

Multiple Platform Development

Because of the proliferation of PC operating environments, commercial products need to support multiple platforms to reach the widest possible audience. This article demonstrates using OS/2 as the development base to build a real product that runs under four different environments. The design and coding techniques and tools necessary to maximize code sharing between products and minimize maintenance costs are explained with examples.

OVERVIEW

When building a PC commercial application, one of the first questions is, What is the target platform? In the past, the answer was obvious: PC/MS DOS. Today, while DOS is still a candidate, developers must also consider OS/2, Windows, and AIX/UNIX. DOS presently has the largest user base, but people are moving away from it. Windows has become very popular with a large user community, but it is being challenged by OS/2. And although OS/2 usage is growing rapidly, it will have to contend with Windows 95 (Chicago). AIX/UNIX will never go away.

One solution is to build for all of them or at least more than one. This approach is practical only if you are able to

write the application once and at build time determine the target platform. Otherwise, you end up writing multiple applications and, more importantly, maintaining multiple applications.

Late last year, Spitfire Software was faced with this dilemma. We needed to upgrade a disk utility we market that targeted DOS and supported only a command line interface. It needed an interactive interface, additional function, and a broader audience. From an interface standpoint, both OS/2 Presentation Manager and Windows were attractive. The command line version was still a requirement for use in batch files and REXX programs. AIX/UNIX would have to wait because of skills. Therefore, the four target environments we chose were:

1. OS/2 Presentation Manager
2. OS/2 command line (non-Presentation Manager)
3. Windows
4. DOS command line.

TOOL SELECTION

Next, we selected the development environment and tools. OS/2 was the logical choice for the development environment, since it also supports the DOS and Windows environments and is a solid operating system. C++ was chosen as

the development language for portability and productivity reasons. The object-oriented nature, tight type checking, and functional richness of available libraries has made C++ a strong application development language.

What we did not want to do was code the interactive interface for OS/2 Presentation Manager and Windows at the C++ level due to productivity concerns. A number of graphical user interface tools on the market allow design of the interface at an object level and generate the code. We considered two: Gpf Professional Developer's Toolkit and JBA International's Guidelines. They both generate C++ code and interface to external C++ modules. We selected Guidelines for a number of reasons, none of which reflect negatively on Gpf. Guidelines provides

basic facilities that significantly increase productivity:

1. It is a fully integrated development environment with workbench-supporting design, implementation, test, and documentation activities.
2. It supports a full set of controls, including CUA '91 in both OS/2 and Windows environments (using IBM CUA Common User Access Controls Library/2 for Windows applications).
3. It supports Borland C++ compilers for both OS/2 and DOS/Windows and WatCom C++ and IBM's C Set++ for OS/2.
4. It supports multiple execution threads in both environments.
5. It supports complete integration of external C++ modules, with the main module allowing as much application function

placed in the external modules as desired. For writing the main module event processing functions, Guidelines provides a proprietary language, JOT, which is similar to Basic in appearances (modified to force structured programming) with C++ data typing added. It allows the addition of in-line C++ statements. JOT is a full-function language in which small- to moderate-size complete applications can be written.

We chose Borland's C++ compiler. Although there was no other choice considering we needed one that Guidelines supported for both OS/2 and DOS/Windows, it was a good selection.

IMPLEMENTATION

The basic design for the project placed the user interface manage-

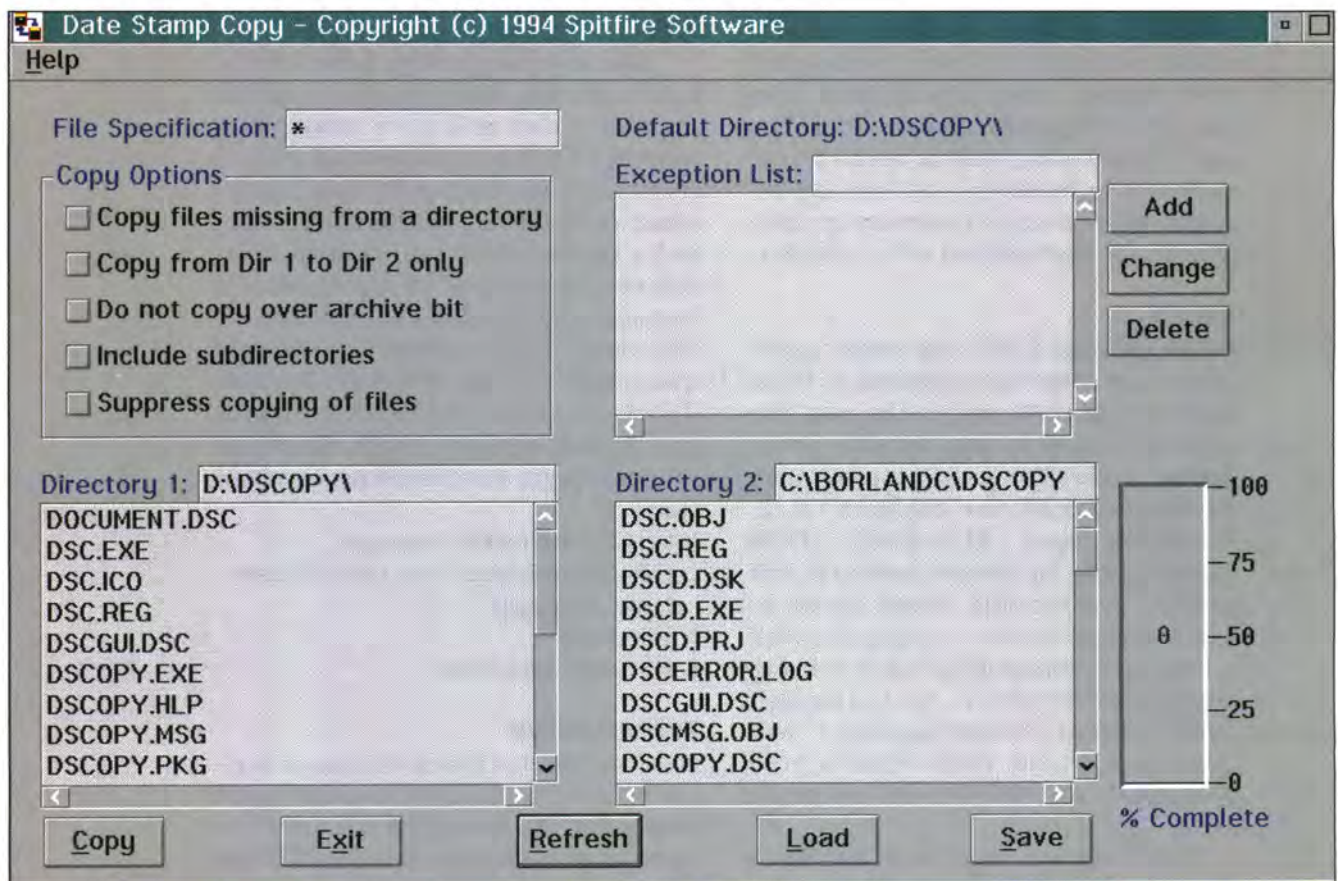


Figure 1. Date stamp copy—OS/2.



ment functions and high-level application logic in the main module and all of the actual application function in separately compiled modules. One source code set was written for the application function modules, and a main module was generated using Guidelines for both OS/2 Presentation Manager and Windows environments. A second main module was written in C++ for the OS/2 and DOS command line versions.

Developing the interactive user interface module with Guidelines was straightforward, and no special consideration was given to OS/2 vs. Windows initially (Figures 1 and 2 show how the interface looked in both environments). We designed and tested the interface under OS/2. When it was complete, we tested it under Windows. At this point, we encountered an irritant. While the

Windows version had all of the components and the code ran correctly, the interface did not look right because proportions and colors were different. We had to go back and fine-tune the interface for Windows. This was difficult because in the design step you see how it looks under OS/2, not Windows. To see it under Windows, you have to build and execute the Windows version.

Implementing the C++ modules was more complex. We ran into the following problems:

1. The operating system header files (OS2.H, WINDOWS.H, and none for DOS) are quite different in specifying operating system data types.
2. The OS/2 and DOS API's are very different. Although the Borland function library hides many of the differences, it does not hide them all. For example,

OS/2 provides an API call to copy a file while DOS does not. Also, OS/2 has both High Performance File System (HPFS) and file allocation table (FAT) file systems, while DOS supports only FAT file systems.

3. Guidelines provides its own string class, which is functionally richer than Borland's and has to be used in the OS/2 Presentation Manager and Windows versions. It could be used in the OS/2 command line version but requires a 386 or higher processor. The DOS command line version needed to support all x86 processors, so we had to use the Borland string class for that version.
4. OS/2 has a flat address space, while DOS/Windows is segmented and requires "memory model" considerations.
5. Borland's OS/2 compiler trans-

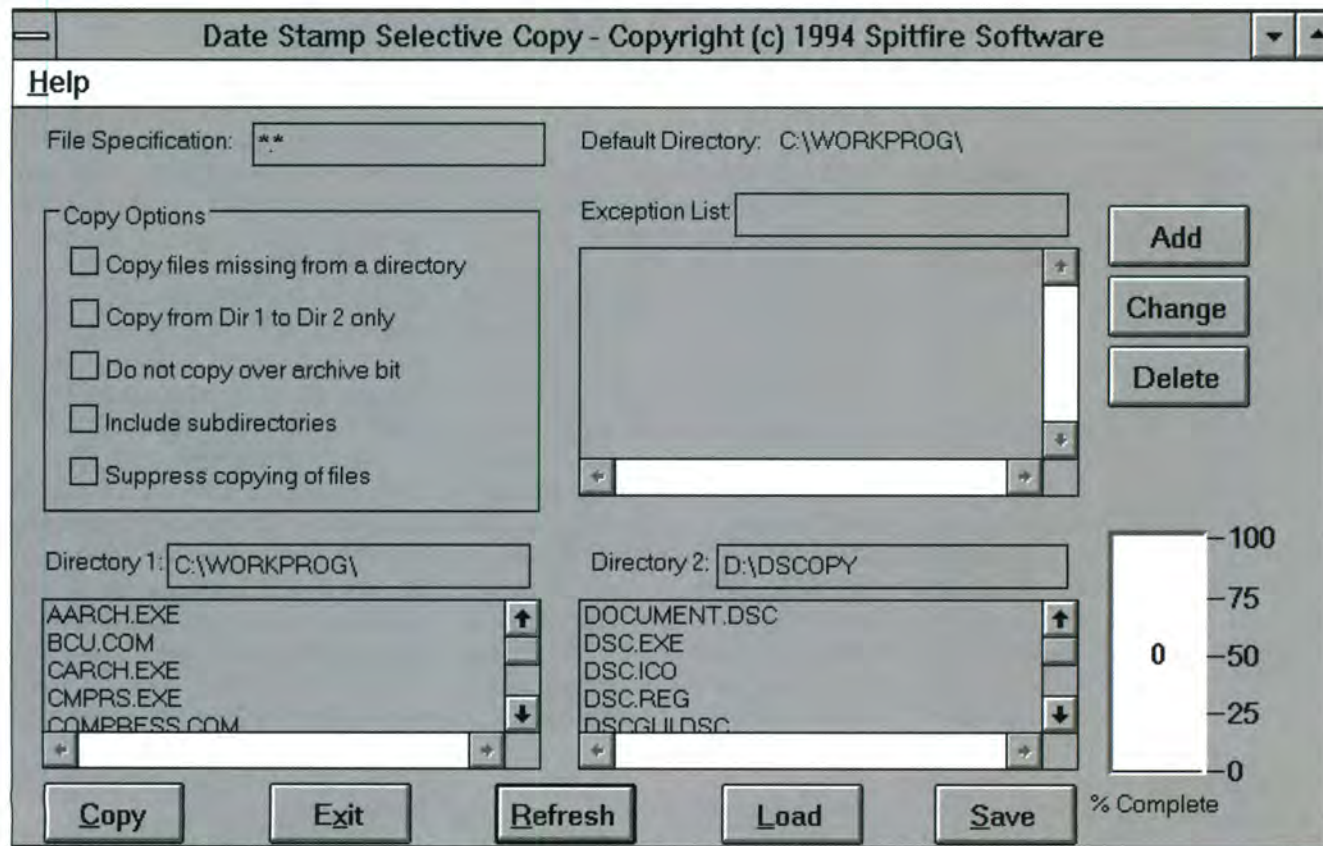


Figure 2. Date stamp copy—Windows.



lates the `int` data type to a long integer type (4 bytes), while the DOS/Windows compiler converts it to a short integer type (2 bytes). This causes a problem using Borland's libraries' data structures within JOT (which supports only short or long integer types and does not recognize the `int` type). For example, the Borland date type structure:

```
struct date {
    int da_year;
    char da_day;
    char da_mon;
};
```

will result in `da_year` being long when compiled for OS/2 and short when compiled for DOS/Windows.

We solved each of the problems. The basic solution to the first four problems is to use the C++ conditional compile directives and macro facilities. To resolve which header files and macro definitions are used to establish the environment, every C++ module had at its beginning the statements in Figure 3.

In Figure 3, `__OS2__` is defined if you are compiling under Borland C++ for OS/2, and `_Windows` is defined if compiling for Windows. Neither is defined when compiling for DOS.

In examining Figure 3, note the following:

1. In the OS/2 section, "guirun.h" is included for access to the Guidelines string class. When working with a third-party class library, it is the header file that supplies the information necessary to use the methods and overloaded operators. The header file "os2.h" was included for two reasons: "guirun.h" requires it, and the module makes direct calls to the OS/2 API. The three `#define` statements set compile time directives in "os2.h" and "guirun.h".

2. In the Windows section, "guiwun.h" is the equivalent of "guirun.h". The header file "windows.h" is included because "guiwun.h" requires it.

3. In the DOS section, "strng.h" is required to use Borland's string class. The two `#define` statements define macros that in the other sections were defined by the Guidelines and operating system headers and are used in the code when referencing variables shared by the main module and the application function modules. Since we do not have the other headers in the DOS section, we needed to define them here.

Originally, we had planned to use the Guidelines string class in all four versions, since it had to be used in the OS/2 Presentation Manager and Windows versions and was functionally richer than Borland's string class. Therefore we made extensive use of that class's methods. For example:

```
GuiStrLen(a);
```

returned the length of a string. This method was used in allocating space for a char array to process the string "a" in a standard C manner. Later, when we discovered we could not use the Guidelines string class for the DOS version because of processor requirements, we were faced with a dilemma. Either we used conditional compile directives throughout the code (selecting either Borland's or Guidelines' methods) or we had to find a "least common denominator" that would work for both. The solution was easy. In the case of retrieving the string length for all versions, we changed the statement to:

```
strlen((const char *) a);
```

Both the Borland and Guidelines

string class provided for casting a string as a constant char array.

To set various limits that depend on a flat vs. segmented address space or the file system being used, the macros in Figure 4 were defined. Borland does include a set of predefined macros for limits in "limits.h" that could have been used for the path and file name size values. The list was incomplete from our standpoint, so we decided to create a list tailored to our needs.

Conditional compile directives were used in the code itself for special cases, as illustrated in Figure 5. The Borland library handles many of the differences in code for the environments, but when directly calling operating system API's, you must handle the differences. For example, the application module that copies a file uses the API function under OS/2 but needs to do the physical copying itself under DOS and Windows.

In Figure 5, the first part defines the common variables used in both environments and is followed by the variables used in only one environment. The code section starts with the code common to both, continues with code specific to one or the other environment, and ends with the common exit code. Also note that the macros in Figure 4 define the size of the buffers used in Figure 5.

After examining Figure 5, it can be argued that what we have is in fact two separate programs built as one but as hard to maintain as two. I would agree if this was the rule, but it is the only place in the application where the code itself needed to be different depending on the environment. In all other places, Borland's compiler handled the differences through its libraries.

How the C++ `int` type is handled is a different problem because



it effects the JOT code, which does not support compile directives and macros. What JOT does is allow us to write in-line C++ code that is passed directly through to the compiler. For example, if we have a variable that is shared with the application C++ modules and defined as an `int`, we can process it as shown in Figure 6 (JOT uses a semicolon to denote the start of a

comment). The one drawback of this approach is that the variable can be referenced by only in-line C++ code.

Although the code fragment in Figure 6 is nonsense, it illustrates adding C++ code to JOT modules. Guidelines does let us create functions that are entirely C++. The problem is that it does not syntax check in-line C++ code at genera-

tion time. It is not until you compile the code that you discover errors. With JOT statements, Guidelines provides very rigorous checking, and if the code generates C++, you know it will compile.

The last problem concerns memory models and processors. The solution here is quite straightforward and handled by the makefiles. One large makefile could not

```
#ifdef __OS2__
#define INCL_BASE
#define INCL_NOPM
#define INCL_DOSFILEMGR
#include <os2.h>
#include "d:\guide\sys\guirun.h"

#ifdef _Windows
#include <windows.h>
#include "d:\guide\sys\guiwun.h"
#else
#include <strng.h>
#define SHORT short
#define LONG long
#endif
#endif
```

// test for OS/2
// OS/2 base/functions
// OS/2 no PM
// OS/2 File manager
// required for guirun.h
// Guidelines OS/2 header
// used for String class #else
// Windows or DOS
// test for Windows
// Windows functions
// Guidelines Win header
// used for String class
// DOS
// use BCC string class
// define macros defined
// in windows.h or os2.h

Figure 3. External C++ module environment statements.

```
#ifdef __OS2__
#define MAXCNT 32000
#define NAMESIZE 256
#define PATHSIZE 260
#define EXCPsize 100
#define SDIRMAX 100
#define MAXBUF 128000
#define MAXPARAMFLE 1000
#else
#define MAXCNT 2000
#define NAMESIZE 13
#define PATHSIZE 100
#define EXCPsize 50
#define SDIRMAX 50
#define MAXBUF 64000
#define MAXPARAMFLE 100
#endif
```

// test for OS/2
// maximum number of files
// maximum file name size
// maximum path size
// maximum number of exceptions
// maximum number of subdirectories
// read/write buffer size
// maximum number of param files

// maximum number of files
// maximum file name size
// maximum path size
// maximum number of exceptions
// maximum number of subdirectories
// read/write buffer size
// maximum number of param files

Figure 4. External C++ module system parameters.


```

/* Routine to copy file fn on path fp to path tp. Input:  File name
                  Source directory
                  Target directory
Return: 0  no error
        -1 target path does not exist
        +1 other disk error
*/
int fcopy(char * fn, char * fp, char * tp)
{
    /* common variables */
    unsigned char * tpn;           // to file name and path
    unsigned char * fpn;           // from file path and name
    int frc = 0;                   // function return code

    /* environment specific variables */
#ifdef __OS2__                     // test for OS/2
        ULONG OpMode;             // OS/2 copy mode parameter
        APIRET rc;                 // OS/2 function return code
#else                             // DOS/Windows environment
        char * buf;               // read/write buffer
        int inh;                  // read handle
        int outh;                 // write handle
        long total;               // file size
        unsigned len;             // length of read/write record
        unsigned fdate;           // file date
        unsigned ftime;           // file time
        unsigned attrib;          // file attribute
#endif

    /* common code */
    /* allocate buffer for source path/filename */
    if (NULL == (tpn = (char *) malloc(MAXPATH + 1)))
    {
        //process error
    }

    . more common code
    .
    /* environment specific code */
#ifdef __OS2__                     // test for OS/2
        OpMode = DCPY_EXISTING;    // setup call to file copy
        rc = DosCopy(fpn, tpn, OpMode); // execute call
        if (rc != 0)               // test for error return
        {
            // process error
        }
        else frc = 0;
#else                             // do physical copy in DOS/Windows environment
        /* allocate disk read/write buffer */
        if (NULL == (buf = (char *) malloc(MAXBUFF)))
        {

```

Figure 5. Example external C++ function (continued on page 37).



```

        // process error
    }

    . Do physical copy of file
    .

    /* clean up and return */
    if (!_dos_getftime(inh,&fdate,&ftime)) _dos_setftime(outh,fdate,ftime); // set file date/time
    free(buf);
    close(inh);
    close(outh);
    if (!_dos_getfileattr(fpn,&attrib))
        _dos_setfileattr(tpn, attrib); // set file attribute
#endif
//end of environment specific code

    /* common exit code */
    if (frc == 0) fprintf(log,"Copied %s to %s\n",fpn,tpn); free(tpn);
    free(fpn);
    return(frc);
}

```

Figure 5. Example external C++ function (continued from page 36).

be used since the OS/2 build needed to take place under OS/2 while the DOS/Windows build required DOS. The Guidelines module required only instructions telling it which environment to build. For the external C++ code, three projects were set up under the Borland IDE: OS/2 (one project handled both Presentation Manager and command line), Windows, and DOS. They all use the same source code but have different compile time options and place their intermediate and final output in different subdirectories. The one small irritant is that a complete build requires the execution of the makefiles for the three Borland projects followed by the two builds of the Guidelines graphical user interface. It would be nice if Guidelines allowed the inclusion of the Borland makefiles in the one it generates.

DOCUMENTATION:

The on-line documentation and context-sensitive help was easy to implement under Guidelines,

which has a complete help text generation facility built into it. During development of the OS/2 Guidelines module, we added the help text as we went, creating hyper text links and organizing it in document form. This way we had both an online document the user could read and print plus context-sensitive help panels.

At generation time, Guidelines generated the complete .IPF file for OS/2. For the Windows version, Guidelines generated the .RTF file converting our IPF tags to RTF tags. We encountered one problem here. Guidelines was able to translate many IPF tags but not all. We therefore had to edit the Windows help text, replacing selected IPF tags with RTF tags. During the compile/link phase, the IPF/RTF files were automatically compiled into the .HLP file and linked to the executable. A great side effect was that we were able to take the .IPF file generated by Guidelines and easily create the printed document shipped with the product.

SUMMARY

As shown, the tools now exist to easily develop products for multiple platforms. Key to doing this is a thorough understanding of each platform and proper planning and design during the initial stage of the project. A strong multiple platform compiler is required, and a multiple platform graphical user interface design tool significantly increases productivity.

Bruce W. Landeck is president of Spitfire Software (325 Breakwater Ridge; Atlanta, Ga. 30328; (404) 257-0187), which develops financial and utility software for multiple PC platforms. Bruce founded Spitfire Software in July 1993 after 30 years of experience with IBM in software development. While at IBM, he held a full range of technical and managerial positions, including lead programmer, architect, product planner, development manager, product manager, and business area manager. He has a B.S. degree in Engineering Science from Purdue University and an M.S. degree in Aerospace Engineering from the University of Arizona.



```

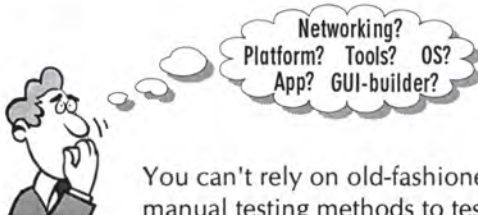
JOT function:    function window.okay.click()
                  ; b is defined externally as a long
                  ; JOT integer gobal variable
                  c: int c = 2; // C++ inline code

                  c: b += c;    // C++ inline code
                  return      ; JOT return
                  end

```

Figure 6. Using C++ in JOT.

If you're developing client/server systems, there's one more question you should be asking: How am I going to test them?



You can't rely on old-fashioned manual testing methods to test complex, client/server applications. You need to automate your testing with a tool designed for client/server. ATF is the tool-of-choice for over 100 corporations building client/server applications under OS/2, Windows, and NT. To learn more about ATF, call 617-576-2257. (Or FAX 617-864-7747.)



The Softbridge
Automated Test Facility

Softbridge, Inc. ▲125 CambridgePark Drive ▲ Cambridge MA 02140

Circle Reader Service Number 12

BARCODE ANYWHERE. for OS/2

Are you seeking ways to increase your productivity and data accuracy while lowering your overall cost of data collection? Look no further than BARCODE ANYWHERE for Automatic Document Indexing Solutions.

Features:

Any Angle
Any Background
Over 60 Pages per Minute
Internal Consistency Checks
Developers Kit also Available
Pop-up Editor for Invalid Barcodes
Optional Zoning for Faster Throughput



Barcode Types

- Code 39
- Code 128
- Patchcode
- Code 2 of 5 4Q94
- EAN Codes 4Q94
- UPC Code 4Q94

Input Sources

- Scanner
- Fax
- Focal Plane Array
- Video Cameras

The only barcode we can not read is one which is pasted on the back side of a page

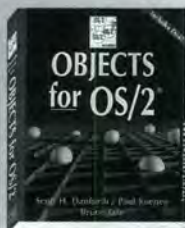
Solution Technology, Inc.

1101 S. Rogers Circle, Building 14, Boca Raton, FL 33487
phone: (407) 241-3210 fax: (407) 997-6518

BARCODE ANYWHERE is a trademark of Solution Technology
OS/2 is a trademark of IBM corporation

Circle Reader Service Number 13

Van Nostrand Reinhold... Your OS/2® Power Source



Object-Oriented Programming Using SOM and DSOM

BY CHRISTINA LAU

This comprehensive guide takes up where the SOM user documentation leaves off, providing easy-to-understand examples and clear instructions that will have first-time users as well as SOM veterans creating SOM- and DSOM-based applications that work. Includes disk.

0-442-01948-3 \$39.95

Lotus Notes® Release 3 in the OS/2® Environment

BY TONY WALSH

What happens when nearly one million users implement the new release of Lotus Notes, the premier client/server Groupware application? System administrators and advanced users in OS/2 will be searching for ways to make the most of its powerful new features. Includes disk.

0-442-01890-8 \$36.95

Objects for OS/2®

BY SCOTT DANFORTH, PAUL KOENEN, AND BRUCE TATE

Integrates the System Object Model with OS/2 client server functions. Explains how OOP and SOM function, why they are important, and how they relate to OS/2. Includes disk.

0-442-01738-3 \$36.95

Running Windows Applications in OS/2®

BY AYODELE ANISE, TERESA BEHR BECK, AND JEAN N. SHORTLEY

How do you make Windows 3.1—and the thousands of applications that run under it—even more powerful? By optimizing Windows to run under the OS/2 environment and taking advantage of the advanced features that only OS/2 can provide.

0-442-01924-6 \$34.95

Dynamic Data Exchange for OS/2® Programmers

BY GLENN PUCHEL

Well-organized and straightforward in its presentation of sometimes complex topics, this book is a valuable addition to the reference collection of all programmers, particularly those whose responsibilities include connecting Windows and OS/2 applications. Includes disk.

0-442-01949-1 \$36.95

OS/2® C++ Class Library

BY KEVIN LEONG, WILLIAM LAW, ROBERT LOVE, HIROSHI TSUJI, AND BRUCE OLSON

This sweeping guide to the User Interface Library is packed with design rationale, tips for designing classes in C++, and tips for programming to the Presentation Manager. Includes disk.

0-442-01795-2 \$36.95

OS/2® and NetWare® Programming

BY LORI GAUTHIER, MORGAN ADAIR, AND WAYNE TAYLOR

The only book to link Netware to OS/2, this excellent reference guide and tutorial offers in-depth chapters on NetWare Client support for OS/2 and NetWare Client SDK. Includes disk.

0-442-01815-0 \$36.95

Developing OS/2® Multimedia Applications

BY WILLIAM W. LAWTON, BRADLEY NOE, AND MARCELO R. LOPEZ, JR.

This programming guide provides readers with an understanding of the overall architecture of MMPM/2, and its subsystems, and shows how to best use the functionality provided by these subsystems. Includes disk.

0-442-01929-7 \$39.95

OS/2® Presentation Manager® GPI, 2nd Ed.

BY GRAHAM C.E. WINN

Application program developers will find comprehensive, up-to-date information on the new GPI features of the OS/2 Presentation Manager as well as countless programming tips, hints and pitfalls to avoid. Includes disk.

0-442-01939-4 \$39.95

Order directly from the Publisher by calling 1-800-842-3636 (Dept. Z1660) or stop by your local bookstore.

Client/Server Survival Guide with OS/2®

by Robert Orfali and Dan Harkey

0-442-01798-7 \$39.95

Client/Server Programming with OS/2® 3rd Ed.

by Robert Orfali and Dan Harkey

0-442-01833-9 \$39.95

Developing C/C++ Software in the OS/2® Environment

by Mitra Gopaul 0-442-01240-3 \$39.95

The OS/2® 2.1 Corporate Programmer's Handbook

by Nora Scholin, Martin Sullivan, and Robin Scragg

0-442-01598-4 \$39.95

The GUI-OOUI War: Windows vs. OS/2®

by Theo Mandel

0-442-01750-2 \$29.95

OS/2® Quick Reference Library Volume 2: Message Functions

by Nora Scholin 0-442-01898-3 \$19.95

OS/2® Rexx Handbook: Basics, Applications, and Tips

by Hallet German 0-442-01734-0 \$34.95

OS/2® Application Programmer's Guide

by Jody Kelly, Craig Swearingen, Dawn Bezviner and Theodore Shrader

0-442-01736-7 \$34.95



Van Nostrand Reinhold
115 Fifth Avenue, New York, NY 10003

1660



*Memory leaks adversely affect your programs' performance and can even cause your machine to stop processing. Here are five ways to avoid and plug memory leaks. By **STEVE HARGIS** and **MIKE SKELTON***

Plug Those Memory Leaks!

All computer programs use memory, some for the program code, some for the data. Some programs call libraries that then use more memory. Data can be held in numerous ways, such as structures, heaps, and stacks. The operating system itself will use memory on behalf of a program. For example, even the mouse pointer shape is held in memory for the program using it. Since program memory can be held by many owners in many ways, determining all the memory that is directly used by a program or indirectly used on its behalf is an involved task. In this article, we define different kinds of memory usage problems, offer some design guidelines to follow during program development, and show an approach to debugging memory leaks.

Since a program uses memory in many ways, in many places, and with many owners, it would not be surprising if some programs lost track of a little bit of memory every now and then. So what? Well, the consequences depend on how much and how fast the program loses track of its memory.

Following are three examples of programs and the consequences of their memory use:

1. A small utility program runs for a few seconds, doesn't keep track of any of

its memory, and relies on the operating system to free its memory when it ends (and hopes no one notices).

2. A business application that runs a few hours needs to seriously track its memory usage to prevent the user from perceiving any affect of lost memory. However, if the lost memory is large enough, the user will notice the application (and the system in which it runs) is becoming slower or encountering errors for mysterious reasons. The user may need to end the program and restart it to make the business application run well again.
3. A mission-critical application that is expected to run 24 hours a day, seven days a week cannot afford to lose track of any memory at all because the entire system will become slow, unresponsive, and incapable of performing at required levels.

All three examples need to manage their memory flawlessly. Plugging memory leaks is an integral part of improving reliability and performance.

MEMORY LEAKS

A memory leak is memory a program allocates (or that is allocated on a program's behalf) that is not freed after the program is finished with it. If the same action is executed again, the program

Fast Visual Application Development for OS/2 and DB2



If you're looking for fast and easy application development for OS/2, then take a look at the award-winning Watcom VX•REXX visual development environment. VX•REXX lets you build applications to exploit the graphical user interface, multi-threading, and multi-processing power of OS/2. VX•REXX Client/Server Edition gives you the added power to access DB2 or other database systems, manipulate the

data, and chart the results at lightning speed.

"We like VX•REXX. Using it for development feels like driving a Porsche: it's fast, it's compact, everything's in the right place, and it makes us look good, too."

Peter Coffee, PC WEEK

Designed to Meet Your Needs.

Watcom VX•REXX combines a project management facility, visual designer and an interactive debugger to deliver a highly productive visual development environment. The Client/Server Edition includes additional powerful objects so you can rapidly create rich GUI database applications. You can create OS/2 client applications which connect to DB2/2 or DB2/6000. Use IBM's DRDA support on OS/2 to access DB2 for MVS, DB2/400 for AS/400, and DB2/VSE and VM (SQL/DS) for VM and VSE. Also supported are Watcom SQL and ODBC-enabled databases[†].

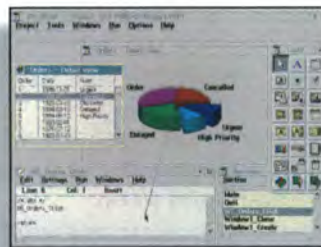
"Overall, this edition of VX•REXX for OS/2 is an outstanding visual client/server development platform." Nicholas Petreley, InfoWorld

- Over 2 dozen objects, including CUA'91 containers, notebooks, pop-up menus and more
- Easy to learn event-driven programming model with complete on-line documentation
- Graphically create CUA'91 Presentation Manager objects, quickly customize their properties, and easily attach REXX procedures
- Integration and control of existing applications through DDE, keystrokes or REXX API's
- Support for professional multi-threaded, multi-windowed and drag-and-drop enabled applications
- Package your application as an EXE or PM macro for royalty-free distribution
- Code reusability through section and file sharing

Point. Click. And Presto!

To create an application you draw user interface objects, customize their properties using standard OS/2 notebooks, and define their event code using powerful drag-and-drop programming. To add database access just draw a query object, visually design a SQL query, press OK and presto— your window is automatically populated with objects that are bound to your query to display, update and search your data.

"Drag-and-drop nirvana." Nicholas Petreley, InfoWorld



Give Your Data a Whole New Image.

Energize your applications by displaying your data in a 3D chart. The Client/Server Edition gives you more than a dozen chart types to choose from, along with over 150 display options. You also get complete support for run-time events so you can bring new drama to your data by making your chart interactive.

"VX•REXX is a must buy." Jacques Surveyer, ComputerWorld

Standard or Client/Server Edition— Which one is for you?

To start creating powerful OS/2 GUI applications right away, order your copy of Watcom VX•REXX Standard Edition for just...\$99*

Or, to start creating rich client/server database applications, order Watcom VX•REXX Client/Server Edition for just...\$299*



1-800-265-4555

Watcom

A Powersoft Company

Watcom International 415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2 Tel. (519) 886-3700 Fax (519) 747-4971 *Prices and specifications are subject to change without notice. Price does not include freight and taxes where applicable. Prices quoted in US dollars. [†] ODBC drivers are available from INTERSOLV, Inc. Watcom, the Lightning Device, and VX•REXX are trademarks of Watcom International Corporation. Other trademarks are properties of their respective owners. ©Copyright 1994 Watcom International Corporation.



allocates more memory instead of using memory previously allocated. With repeated executions, the program accumulates more and more memory. Negative affects of memory leaks include the accumulation of memory by the leaking program causing other processes' code or data to be swapped or paged out. Soon, the system's resources are consumed while the operating system swaps memory in and out instead of executing application code. This robs every process in the system of performance.

MEMORY OVERWRITE/ MEMORY WALKER

A memory overwrite occurs when a program allocates less memory than it actually writes. The effect is to overwrite and possibly wipe out data that happens to be located immediately after it in memory. Whether or not a memory overwrite occurs depends on how many additional bytes are stored beyond the amount allocated. The observable symptom is intermittent data corruption, which is dependent upon the length of data stored in memory (not an intuitive item to check).

MEMORY NEUTRALITY

This is the desired memory use quality of programs. A module, component, function, API, program, system, and so on is said to be memory neutral if it:

- Deallocates all memory for which it is responsible
- Does not attempt to deallocate more memory than that for which it is responsible.

The basic definition is straightforward: deallocate all and only the memory that you allocate. However, when specification standards call for a different module to deallocate memory than the module that allocated the memory, the definition gets a little fuzzy. This is not a recommended specification.

Now that we know what memory leaks are and how they affect the performance of an application as well as each process executing on a system, five recommendations follow that allow developers to detect and fix leaks early in the product development process.

DESIGN AND PROGRAMMING RECOMMENDATIONS

1. Know the memory usage of each component within the product.

It is difficult to diagnose memory problems when no one knows how the product is supposed to use memory. The logical person to know the memory usage is the developer, who possesses some basic memory debugging skills and tools.

To check your knowledge of the memory usage of your application, select a test case that is typical for a user. Assume that you are going to execute the test case while your application is executing. How much memory do you think will be allocated during the test case? How much memory will be committed? Do not forget to specify whether the memory will come from the heap or be part of shared memory, and so on.

Check your answers with a memory analyzer, such as Theseus2 (part of the SPM/2 product).¹ With Theseus2, from the initial window, select the process executing your application, then choose the Memory Utilization option from the Process item on the menu. When compared with results before the test and results during the test case, the Total System, Total Shared originated, and Total Private rows will allow you to check your answers. (It may be desirable to explore memory further with Theseus2 to see exactly what memory was allocated by whom, what the contents are of some memory, or why the size al-

located is not the same as what your application asked for.)

2. Do not rely on the operating system to delete memory when the process dies.

This is common practice with, it would seem, no ill effects. It is true that the operating system will deallocate resources when the process dies, but that works only for relatively short-lived processes. Major processes that stay up for days, weeks, or months should not be implemented that way. There are five main reasons 24/7 processes and, arguably, even shorter-lived processes should not rely on the operating system to delete resources.

First, resource consumption will grow over time, and the operating system may not clean up memory structures in a timely manner. For example, if memory is allocated to store information until a process dies and the process does not die for three weeks, the information stored will consume a lot of memory. Consuming a lot of memory will cause memory (wherever it is used the least) to be swapped. It will not take long for the swapper file to be so large that it adversely affects the performance of everything that runs on the machine.

The second reason is that the code may be ported or expected to run on a different operating system or platform with different clean-up characteristics. The new platform may not clean up resources as nicely as OS/2. Or perhaps the other platform does clean up nicely today, but it may not in the next release.

Reason three relates to device drivers. Device drivers can, and many do, allocate locked and resident memory on behalf of an application. Unless this memory is deallocated by the device driver, it will always be allocated.

A fourth reason to not rely on the operating system to delete

memory when a process dies is that program design changes may alter the operating system's ability to clean up memory structures. Code may originally be written to run in a process of its own, but the process or thread model of products is subject to change. Code may be consolidated from several processes into a smaller number of longer-lived processes to avoid process creation and deletion overhead. Therefore, code that started out in a process of its own may be consolidated to run on threads of a process containing other threads. Resources such as memory, semaphores, and files are owned by the process, not by the thread. So when the code now terminates the thread, the process remains—along with all the resources accumulated by the thread but never freed.

The last reason is that debugging is greatly complicated by not knowing whether memory is supposed to be unallocated by the operating system or if a memory leak exists. We advocate use of the C Set/2 and C/C++ debug memory functions.² When using them, we have noticed that many programs are not memory neutral at the instruction immediately prior to returning control to the operating system. So if we are trying to fix a memory leak, we must first get through the "noise" of memory that is supposed to be cleaned up by the operating system (we say that memory was leaked purposefully) to memory that is being leaked accidentally. This is an extremely time-consuming activity, especially if the developer does not know the memory usage of his or her component.

Let us close this recommendation by saying that sound coding practices include allocating machine resources as needed, tracking those resources, and deallocating them at the earliest reasonable time. Relying on the operating sys-

tem to clean up a process's use of resources is dangerous in today's cross-platform code environment.

3. *Avoid designs where one module allocates memory and another module deallocates it.*

The exception, of course, is when the design specification comes from a standards institute

and you have no control over it. This sort of design can lead to the following problems:

- Poor communication between memory allocation and deallocation modules, or developers, will lead to memory leaks.
- There is an increased possibility of several modules using different



The secret's out!



- Native OS/2 spreadsheet •
- Objects to the core • REXX scripting •
- Small • Fast • Powerful •

MESA™ 2
for OS/2®



The spreadsheet that's more than the sum of its parts.

**TO ORDER: CALL
1.800.315.MESA**



ATHENA DESIGN
SPREADSHEET EXCELLENCE

17 St. Mary's Court
Boston, MA 02146
phone 1.617.734.6372
fax 1.617.734.1130



linking conventions (for example, optlink and syslink). This causes applications to leak memory since memory allocated under one linking convention may not be recognized under another.

- Few people, if any, will really know who is allocating what for what or who can deallocate or

reuse what under what conditions. Communicating that to team members will be extremely difficult, as will answering and debugging memory usage questions.

4. *Minimize the number of different compilers used in the product.*

Using multiple C run-time libraries to compile different mod-

ules will lead to memory leaks because run-time libraries do not implicitly communicate with one another. A corollary to this suggestion is that different versions of the same compiler often have the same problem because the header files used to make the libraries are not the same between versions. It is extremely difficult, if not impossible, to cleanly separate functions within modules or modules within applications among different C run-time libraries. They inevitably will want to communicate, but compilers do not support communications across run-time libraries.

5. *Set the criteria for determining memory neutrality early in the development cycle.*

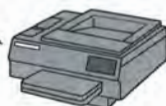
We recommend that memory neutrality be a test exit criteria beginning with the functional verification test. If an application has memory problems, finding and fixing them as soon as possible in as small an environment as possible is much easier than waiting for tests in a large, complex environment.

DEBUGGING TECHNIQUES AND TOOLS

OS/2 memory usage includes these three facts: when a program allocates memory, it acquires a memory object; memory is owned by the process that allocated it; and a program may have multiple threads of execution in a process. So a memory object allocated by any thread is owned by the process and may be shared between all of the threads in the process.

When debugging, the first question to ask is, Which of the many processes running in the system have memory leaks? Once the leaking processes have been determined, the next question is, Which memory objects in the processes have memory leaks? The last question to ask is, What piece of code is allocating the memory objects and what should be deallocating them?

SINGLE KEY-STROKE SCREEN PRINTING



CLIPBOARD VIEWER



SCREEN SAVER

SCREEN CAPTURE
.WPG .MET .BMP .PCX .TIF .GIF .PCT



- Quality Color or Black & White copies of color Desktop Images.
- Copy any portion of any image on the Desktop.
- Screen Saver prevents monitor burn-in.
- Date & Time Display for Time-Stamping images.
- LAN installable. Complete support for LAN attached printers.
- Economical! 4 Utilities for 1 Price! Entity licensing available.
- The most stable and reliable product available.
- Includes both OS/2 2.1/2.0 (32-bit) and OS/2 1.3 (16-bit) versions.

THE LEADER: **PrntScrn**TM Screen Image Utility for OS/2



TM

MITNOR Software

Phone: (918) 357-1628

Fax: (918) 357-2869

SCREEN IMAGES IN OS/2 DEVELOPER ARTICLES ARE CAPTURED USING PRNTSCRN, COURTESY OF MITNOR SOFTWARE

Circle Reader Service Number 17

In the remainder of the article, we describe how to discover if your application is leaking memory. If you do have a memory leak, our detection technique will quickly indicate which part of your application is responsible. Useful tools to complement our technique are also discussed. The first necessary tool is a memory analyzer, which opens access to the entire contents of memory. The second is part of the IBM C Set/2 or IBM C/C++ compiler.³ Other compilers or third-party add-on products offer similar capabilities. Detailed discussions and examples using these tools are available.⁴

Depending on the type of leak, you can use either a memory analyzer or compiler tools to diagnose the cause of the memory leak. Since leak detection begins with a memory analyzer, we will cover them before we cover compiler or compiler add-on tools that are helpful.

MEMORY ANALYZERS

Maneuvering through the labyrinth of OS/2 memory structures is a chore that few developers try to complete. Fortunately, tools exist that will navigate through OS/2 memory and report what is occurring. Memory analysis is made possible by tools that access system and kernel memory; report on process and memory object ownership; understand linear and physical memory addressing; can access shared, swapped, and present memory; and detect memory leaks, among many other capabilities. Theseus2 is the most complete OS/2 memory analysis tool available; its documentation and online help contain years of memory debugging experience and advice (including an "Explanation of the contents of this window" menu choice for every window). A tutorial for the memory leak detection functions of Theseus2 is available.⁵

To confirm that a memory leak

exists, use the system level memory leak detection function in Theseus2. When a memory leak exists, Theseus2 will report that a process (or processes) is not deallocating all of the memory that was allocated. (Focus your attention on the processes that you can start after the leak detection function

has started and the processes that complete before the leak detection function has completed. Thus you would usually ignore system and Presentation Manager processes.)

By confirming the existence of a memory leak, we have also discovered which process is leaking (this is called system level leak de-



OnCmd® xBase for OS/2 ... A Winning Combination!

OnCmd harnesses the power and the speed of OS/2, enabling you to develop new applications or convert existing xBase applications, such as those developed in FoxPro®, Clipper®, and dBase®.



Preserving Your xBase Investment

OnCmd can quickly convert your xBase applications to run directly in OS/2's 32 bit Presentation Manager. This minimizes development time and preserves your current xBase investment.

Text applications are automatically converted to a native. Presentation Manager application that is both multi-user and network capable. With simple code changes, you can add buttons, check boxes, and drop-down menus to give your application more functionality.

Applications Development

OnCmd's screen painter, client-server facilities, "event driven" programming, and Dynamic Link Libraries (DLL) all add to the development power that you need.

A Performance Winner

OnCmd is a performance winner!

No windows emulation overhead.

Disk requirements less than 1 MB.

Installation under 5

minutes. In addition, it typically indexes large databases twice as fast as the competition in half the disk space. For more benchmark details, call or email us.

Client/Server Ready

OnCmd allows you to create a client server environment without the need for a dedicated server system. You can

evolve into the world of client/server at your own speed and at a very low cost, allowing you to take advantage of the improved performance and application stability of client/server.



*Go for the gold.
Order your
copy today!*

ON-LINE

ON-LINE DATA

5 Hill Street, P.O. Box 65, Kitchener, Ontario, Canada N2G 3X4

Tel: (519) 579-3930 Fax: (519) 579-2130

CompuServe: 70022,104 Internet: oncmd@onlinedata.com

On-Line Data recognizes the following trademarks: FoxPro (Microsoft Corporation), dBase (Borland International Inc.), Clipper (Computer Associates International Inc.), OnCmd (Online Data).

Circle Reader Service Number 18



tection). With this knowledge, we would apply the memory analyzer to only the process that is leaking to discover which memory objects are leaking (this is called process level leak detection). In Theseus2, this would require use of the "Memory Object Leak Detection" function, which monitors a specific object in memory and reports the status of each page within the object (this is part of object level leak detection). As time progresses, changes in the status of a page are reported along with a detailed description of what changed at what memory address. This information is extremely helpful because we can learn:

- The size of the memory object
- The contents of the memory object
- The status of each page within the memory object.

Theseus2 allows leak detection in a manner that continually narrows the search. The results from system level leak detection indicate a process or processes; the process level indicates a memory object or several objects; and object level leak detection indicates a page or set of pages. Each level of leak detection points to a lower level of memory abstraction until memory pages are reached.

Armed with this information, we can find the developer responsible for using that memory object and determine if the behavior we are seeing is expected. It can be helpful to ask questions such as, Do you recognize the data in this memory structure? What size should this memory structure be? Is the size of this memory structure static or dynamic? When was this

memory structure allocated and when is the earliest possible time that it can be deallocated? If the developer can answer these questions, then either the design included a memory leak (that is, the developer never planned to deallocate the memory) or comparing the answers to the code will uncover a discrepancy between the design and the implementation. Either way, the memory leak will be obvious.

If the developer cannot answer the questions, you may have to continue your debugging using the compiler tools discussed ahead. You may also want to refer that developer to the first design and programming recommendation at the beginning of this article.

A useful memory analyzer will allow you to analyze memory from several perspectives. Depending



OS/2™ is a great operating system - except when it comes to delivering a consistent desktop time after time. It's so easy to drag and drop system files and utilities into the shredder. Anyone can do it. Now that's an administrative nightmare.

Unless you have the **Desktop Observatory™** from Pinnacle Technology. The Desktop Observatory silently takes control of your user desktops.

Drag and Stop.

Lock & Load...Over a LAN!

Drag the OS/2 System folder into the shredder? No problem. The Desktop Observatory can dynamically rebuild their desktop from a small configuration file you control, and everything is restored. Store the desktop configurations on your LAN, and you can instantly update hundreds of desktops from a central location.

Security & Auditing

But there is even more to the Desktop Observatory. Apply restriction settings to objects so that they cannot be moved, copied, deleted, renamed, or even seen. Password protect folders and programs, and restrict access to files, folders and directories.

You can even start REXX or C utilities when folders or programs are opened or closed - great for backups, statistics collection, audit control, and copying files to or from a server.

Now you've found the perfect solution for OS/2 security, management and control - the **Desktop Observatory for OS/2.**

1-800-525-1650



Desktop Observatory is a trademark of Pinnacle Technology. OS/2 is a trademark of IBM Corporation. ©1994 Pinnacle Technology • P.O. Box 128 • Kirklin, IN 46050 • 317.279.5157

on how much you know about the memory leak at hand, analyzing at the system, process, thread, executable, or memory object level can be helpful. Being able to monitor swapper file growth and page swapping is also helpful in determining if a memory leak exists. We have used Theseus2 to detect and solve many memory leaks by first determining which process is leaking and then examining that process in finer granularity until the actual memory object being leaked is found. Once you know the module or memory object that is leaking, you can apply the compiler debug memory tools if necessary.

COMPILER TOOLS

The IBM C/C++ Tools compiler contains a battery of debug memory management features. These features consist of debug versions

of the `calloc`, `free`, `_heapmin`, `malloc` and `realloc` functions. The debug versions are, respectively, `_debug_calloc`, `_debug_free`, `_debug_heapmin`, `_debug_malloc`, and `_debug_realloc`. There is also a `_dump_allocated` function that outputs information about every memory object on the heap. (Version 2 of the compiler also supports debugging the new and delete operators.)

The compiler can be told to use the debug versions of the memory management functions instead of the regular versions at compile time with a compiler switch. When you turn the switch on, each memory function call automatically invokes its debug equivalent and another debug memory function called `_heap_check`. `_heap_check` performs a thorough check of the heap, its memory objects, and all associated pointers. If

anything is out of order, processing is aborted and a diagnostic message is output on `stderr`. At first it may be annoying that processing is aborted, but it is helpful to know that memory was overwritten or that a bad pointer was passed to a memory function.

When a program aborts due to memory errors, it is common for the run time to indicate that the error was before or after module `x` line `y`. An error message like this indicates that an error was just detected; therefore, the actual error occurred between the line referenced in the message and the last memory function called (specifically, the last `_heap_check` performed). Finding the memory function that executed just prior to the one flagged is usually easy, but it can be quite difficult in a multi-threaded environment. The hunt is



The Award Winning, Multi-Function Enhancements for OS/2®



DeskMan/2™

DeskMan/2™ Workplace Shell™ Manager

Efficiently Manage Your Workplace Shell!

- Easy to use drag & drop interface
- Complete command line interface + REXX & C/D support
- Manage standardized or personalized desktops, customizing access to features
- Selectively save and recreate objects; even transport them between desktops, computers & versions of OS/2
- Recover from or prevent accidental changes to your desktop
- Much more - total management of the Workplace Shell

VUEMan/2™ Virtual Desktop Manager

Increase Your Productivity!

- Makes multitasking easy; use and organize many concurrent activities with an easy to use drag & drop, point & click user interface
- Switch effortlessly between as many as 81 "virtual desktops" - like having 81 monitors that fit right on your desk!
- Drag and drop objects from WPS to virtual desktops, and between virtual desktops
- "Layouts" record window position and size for automatic repositioning and re-sizing at any future time
- Password protection for windows



All this functionality & much more Four Great Solutions - One Low Price!

"DeskMan/2 dramatically improves the ability of corporations and users to get the most out of OS/2. DeskMan/2 helps administrators restrict and control what users are doing and provides robust backup/restore features and seamlessly enhances the features of the WPS."

Dr. Mike Kogan:

Lead Architect of OS/2 2.0
Author: The Design of OS/2



DM2Image™ Configuration Snapshot Facility

Saves Your Working Environment!

- Save or restore an unlimited number of complete system configurations (including active or inactive WPS desktops)
- Terrific for disaster recovery; training & demo machines: other uses
- Extensible rules let you customize configuration definitions
- Configuration snapshots are compressed for maximum efficiency
- Protects WPS desktop, `config.sys`, `startup.cmd`, `.INI` files, `autoexec.bat`, `WIN-OS/2` desktop & system `config.data`

DeskMan/2 Extensions Workplace Shell Extensions

Configure the Workplace Shell to Work Your Way!

- Modify and enhance the runtime behavior of the Workplace Shell
- Enjoy drag & drop without using special keys to shadow or copy
- Control, selectively or globally, all WPS defined object styles, menu items and capabilities - the ability to Arrange, Sort, Delete, Rename, Settings, Lockup, Shutdown, etc. (even object visibility!) is completely definable
- Corporate Edition available with password protection for objects, audit trailing for WPS activities, and other custom enhancements

Copyright 1994 Development Technologies, Inc. All rights reserved. DeskMan, DeskMan/2, VUEMan, VUEMan/2 and DM2Image are trademarks of Development Technologies, Inc. CompuServe is a registered trademark of CompuServe, Inc. Workplace Shell is a trademark of International Business Machines Corporation. IBM, OS/2 and TalkLink are registered trademarks of International Business Machines Corporation.

DevTech

Development Technologies, Inc.
Software Development & Technology Transfer
308 Springwood Road, Forest Acres, SC 29026-2113
Voice: (803) 790-9230 Fax: (803) 738-0218



worthwhile, however, since there are many benefits to letting the compiler find memory errors.

The compiler is the definitive source as to whether or not a module has memory leaks. A home-grown memory check scheme may suffer from the same blind spots as the code it was intended to check. We know of no better way to ensure that the memory management functions are used in a memory neutral way than to use a compiler's debug memory management features. Some memory errors are detected only when the memory debug functions are turned on (for example, passing a bad pointer to free and some memory overwrites).

Generally, compiler tools are not hard to use; although some code additions may be required for C++ code that redefine or overload the new and delete operators. We recommend locating the point in your code where you return control to the operating system (your code should be memory neutral here). Just prior to this point, add a call to the `_dump_allocated` function that is conditional based on a flag the compiler turns on when the debug memory management calls are used. For example:

```
#ifdef __DEBUG_ALLOC__
    _dump_allocated(16);
#endif
```

It can also be useful to use `_dump_allocated` before and after a function, operation, or test case is executed to determine if the function, operation, or test case is actually memory neutral.

After compiling with the debug memory options turned on, your application may behave differently. It will definitely execute more slowly—approximately double previous execution times. When the heap is in a good state, you should be able to run a test case and then shut down your ap-

plication. Just before control is returned to the operating system, the `_dump_allocated` call will print information about heap storage that is still allocated. Ideally, it will say that your application has no storage allocated. If you do have storage allocated, then you have a memory leak! The report is sent to `stderr` and includes the line number and name of the module that allocated the memory, the pointer that points to the memory, how much memory was allocated, and the first 16 bytes of that memory location (the parameter 16 can be changed to as much or as little as is helpful to you).

SUMMARY

Customers are demanding applications that can maintain peak performance for weeks or months without needing to be restarted. Even small memory leaks will cause an application to grind the entire system to a halt; so detecting and plugging memory leaks is an important part of performance.

Compiler debug memory features and memory analyzers are indispensable tools in avoiding and plugging memory leaks. Memory analyzers allow memory leaks to be quickly detected; they

can also be used to narrow the hunt to specific modules or memory objects. Compiler debug memory tools can then be used to monitor the behavior of specific modules and to flag incorrect memory usage.

Nothing beats having the right tools for the job!

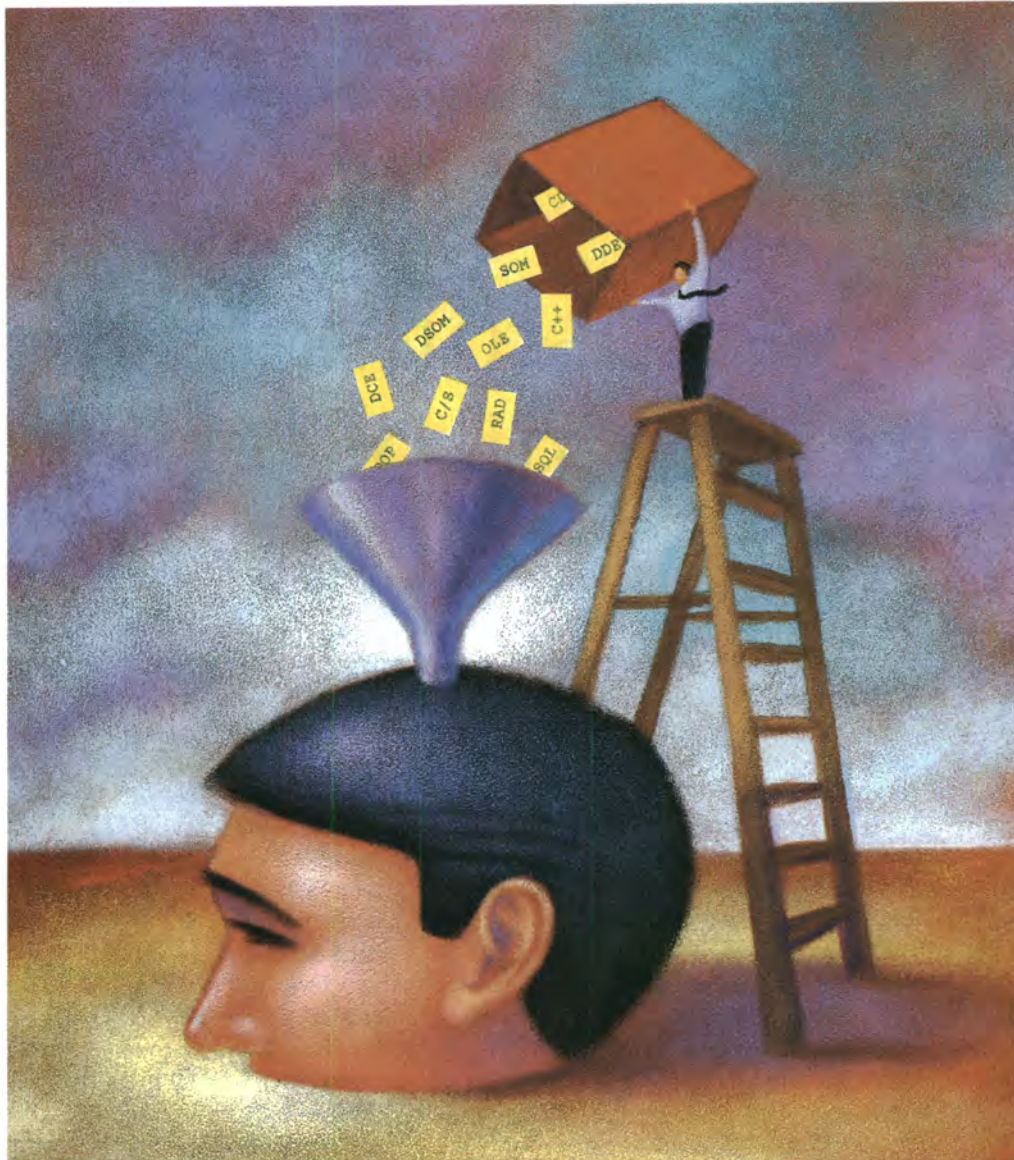
Steve Hargis joined IBM Austin in 1991 to work on the Parallel Database Manager project. He subsequently joined the LAN Systems Performance Group. Steve has an M.S. in database theory from Arizona State University and a Ph.D. in information systems from The University of Texas in Austin. He has authored one book about qualitative, computer-supported negotiations and written several articles on performance topics. Steve can be reached via Internet at hargis@vnet.ibm.com or by phone at (512) 838-0560.

Mike Skelton joined IBM in 1977 after receiving a B.S.E.E. and M.S.E. from the University of Texas at Austin. Mike has designed LSI chips, written device drivers and operating system extensions, and analyzed system performance. Presently, he is a system performance analyst for OS/2 LAN System products. He has published numerous papers about computer algorithms, performance, and performance tools.

REFERENCES

1. IBM System Performance Monitor/2 2.0. (Software available from 1-800-IBM-CALL.)
2. IBM C/C++ Tools 2.00 C Library Reference, 2nd edition, March 1993, pp. 42-44 and under the specific function names. (IBM order number: S61G-1183-00.)
3. IBM C/C++ Compiler 2.01. (Software available from 1-800-IBM-CALL.)
4. Steve Hargis, Mike Skelton. Memory Debugging for C and C++ Programs, IBM White Paper, LAN Systems Performance, March 1994. (Available in PostScript on CompuServe in section OS2DF2 in library 9 as `memlks.exe` (self-extracting file into `memleaks.ps`).)
5. Steve Hargis, Mike Skelton. Memory Leaks, IBM White Paper, The Developer Connection for LAN Systems, Vol. 1, September 1994. (Available from 1-800-6DEVCON.)

Attend the only show that has all the tools you need to make your development vision a reality.



Your mind is your livelihood. And in a world of converging technologies, you need to fill it with more than just the buzzwords. New technologies are supposed to solve your development problems. Yet for every problem solved, *you* know a new one will emerge.

Software Development '95 takes you beyond the buzzwords—from component to full-scale enterprise application development, we show you how to create tangible, real-world solutions.

We explore the issues surrounding object-oriented

programming, client/server migration, rapid application development, and more. You get a clear picture of where the technology is—and where it's headed. SD '95 delivers all the information, tools, and business contacts you need to thrive in today's challenging development world.

Don't miss out. Bring your vision to SD '95. And leave with the tools to make it a reality.

For more information, call (800) 441-8826, or (415) 688-4345 for fax-back info, or e-mail us at sd95west@mfi.com.



Software Development '95, February 13-17, Moscone Convention Center, San Francisco, CA

250 CLASSES • 250 VENDORS • 15,000 DEVELOPMENT PROFESSIONALS

OS2D



*Toolbars are becoming the norm for all types of applications. This article presents a custom Presentation Manager control that saves developers a large amount of the effort and code maintenance required to support a tool bar. By **MARK McMILLAN, GENNARO CUOMO, JÜRIG VON KÄNEL, and GUILLAUME LE STUM***

A User-Customizable Toolbar Control

The graphical toolbar is a popular concept used to improve the usability of graphical-user-interface-based applications. As applications suffer from expanding "featuritis," searching for commonly used functions in a text-based menu bar can become quite tedious. A toolbar gives a user the ability to bypass the text menu bar and have direct access to the most commonly used functions. Its graphical and intuitive visual appearance gives the application a sophisticated look and feel. Toolbars have almost become the norm for even moderately complex applications.

For the Presentation Manager developer looking to provide this type of function, the task is daunting, since Presentation Manager contains no native support for this kind of graphical control. The graphical requirements alone are a significant programming effort. Add to that the code for user customization of the toolbar, and it adds up to a significant project in itself. All of this is overhead to the essential functions of the application.

We present here a custom Presentation Manager control that sports a number of different styles and arrangements of bitmaps to produce a variety of visual effects (as illustrated in Figure 1). By using this control, a Presentation

Manager developer is saved the large development effort and code maintenance required to support a toolbar. This, in turn, allows more development resource for basic application functionality and a shortened development cycle. No expensive graphical user interface builder tools or class libraries are required—just an OS/2 2.x compiler, the standard OS/2 toolkit, and the run-time code supplied in the package. The run-time code may be statically or dynamically linked.

The control was originally conceived at the IBM Yorktown Research laboratory by Gennaro Cuomo and Jürg von Känel for use in EPM (the OS/2 enhanced system editor). Its value for general application development was recognized, and the design was generalized into a prepackaged custom Presentation Manager control, with subsequent development contributions from Guillaume Le Stum, John Ponzio, Alan Warren, and Alex Bertrand. Mark McMillan of IBM Research Triangle Park developed the sample code and frame control utility functions and initiated improvements in the architecture. An earlier version of this control appears on the OS/2 Developer Connection CD Volume 4. The current version contains many improvements and new samples.



Aside from the good graphical presentation, the most significant features of this control are the easy user customization methods it provides (thus the name, UCMenus). Customizing a UCMenus is made easy for the user by extensive use of direct manipulation techniques. For example, to delete an item from the toolbar, the user can drag the item to the system shredder. Re-ordering the items is as easy as dragging the item to a new position and dropping it. Table 1 shows all the direct manipulation functions provided by UCMenus.

Clicking the context mouse button on a UCMenus toolbar pops up a con-

text menu that allows the toolbar styles to be changed and items to be added and deleted (as shown in Figure 2). The control has built-in dialogues that allow the user to change the bitmaps, text, and application function associated with any menu item. Figure 3 shows the notebook used to associate the menu item with an application function.

All customization is encapsulated in the UCMenus control itself. The control and the application cooperate via messages to provide application-specific information during customization. Using the built-in customization methods the user can:



Figure 1. A standard frame window with four UCMenus toolbars.

- Modify styles related to visual effects
- Add and delete items and sub-menu items
- Replace bitmaps with .BMP files or application-supplied bitmaps
- Edit the text under the bitmap
- Associate application functions with menu items (the application supplying the list of valid functions)
- Move/copy menu items within a menu or between menus
- Restore the entire menu to an

application-defined default state.

The application can restrict what customization functions are allowed, which we will discuss later in this article.

UCMENU STYLES

Like many Presentation Manager controls, the visual appearance of a UCMenu can be modified through the use of "style" flags. Figure 1 shows an application with toolbars of several different styles.

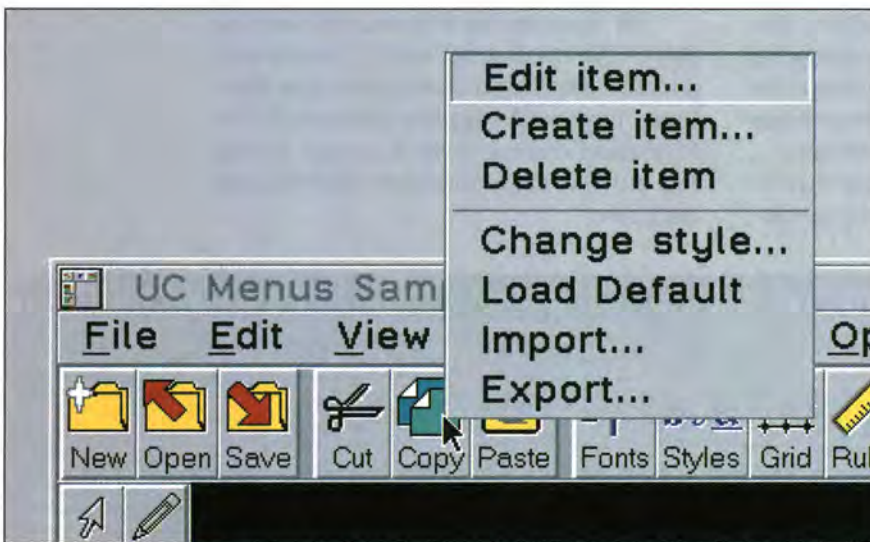


Figure 2. The built-in UCMenus context menu.

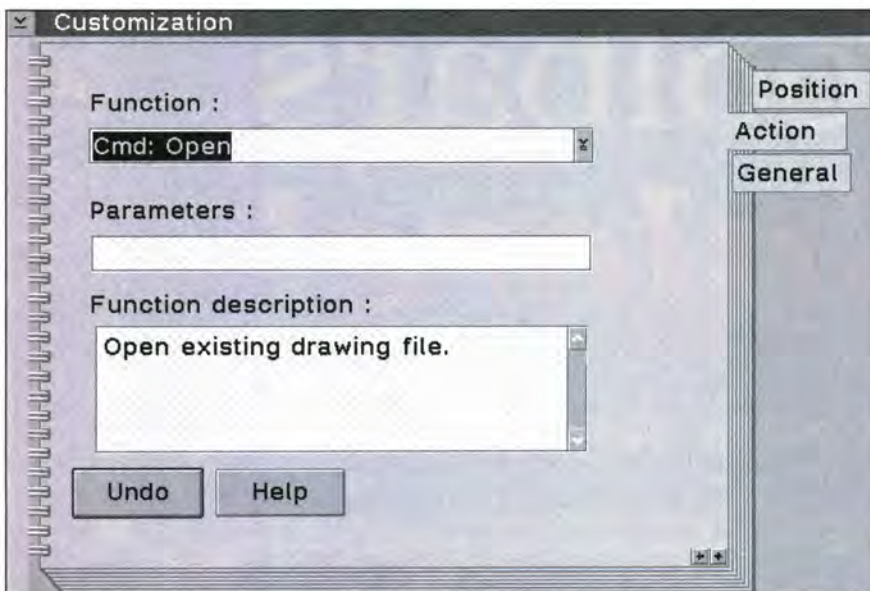


Figure 3. The built-in UCMenus customization notebook.

UCMenu toolbars can have a vertical, horizontal, or matrix layout. The bitmap items can be drawn with or without three-dimensional frames, text underneath, and fixed or automatic sizing. Individual items can be highlighted with a check-mark attribute and are drawn with thick dark borders. Background colors of the menu items and empty toolbar spaces can also be controlled. When required by window size, scrollbar controls will automatically appear at each end of the toolbar.

UCMENU PROGRAMMING CONCEPTS

Several key concepts need to be understood to make effective use of UCMenu controls in an application program. As its name implies, the basic paradigm of a UCMenu control is that of a normal Presentation Manager text menu; the control contains selectable items, submenus, separators, item attributes, and so on. (The current implementation is limited to a single level of submenus.) As we will see, the developer's view of this control is also similar to that of a Presentation Manager menu control. This similarity is by design and allows anyone familiar with Presentation Manager menus to grasp the terminology, concepts, and messages involved with UCMenu controls. It is important to recognize that there are some key differences, many of which we will cover in this article.

The UCMenu control uses many of the same menu-related messages and data structures as its Presentation Manager menu counterpart. (In fact, a real Presentation Manager menu is used "under the covers" of the UCMenu control, but that is hidden from the application.) For example, a `MM_QUERYITEM` message can be sent to the UCMenu window to obtain a `MENUITEM` structure that describes a particular item in the menu. The ap-



plication will receive all the standard Presentation Manager menu notification messages except `WM_DRAWITEM` and `WM_MEASUREITEM`, which are used internally by the UCMenu, and `WM_COMMAND`, which is replaced with a `WM_CONTROL` message.

Because of its added visual and interactive complexity, the standard Presentation Manager menu structures and messages are not sufficient to implement a customizable graphical menu bar. Thus, UCMenus adds another layer of messages and data structures on top of the Presentation Manager menu architecture. It is important to understand the new concepts that UCMenus brings to the existing menu architecture and how the Presentation Manager menu architecture has been extended to accommodate them.

UCMenu Action Strings. One of the most important new concepts that UCMenus brings to a menu is that of *actions*. An action is a textual description of the function assigned to a particular menu item. It is an arbitrary string defined by the application and seen by the user when customizing a UCMenu. It is usually a short two- or three-word phrase that defines the application-specific function the menu item is to perform when selected. For example, a text editor might define action strings like:

"Copy to Clipboard"
"Paste from Clipboard"
"Left Justify"
"Select Fonts"

When creating a new UCMenu item or modifying an existing item, a user will select from a list of actions supplied by the application.

Applications that use Presentation Manager menus often associate specific application functions with specific item IDs. Figure 4 shows the typical processing of a

`WM_COMMAND` message from a Presentation Manager menu control. The application function selected in the switch statement is based on the item ID supplied by the menu control in `mp1`.

For a UCMenu control, the item IDs are arbitrary and can change during execution of the program. An application cannot know ahead of time what the item IDs are going to be for any particular menu item. To understand why, consider what happens when a user adds a new menu item to an existing UCMenu toolbar. The UCMenu control supplies all the user interface necessary for the user to choose a bitmap, text, and an application-specific action for the new item. When the new item is added to the menu, the UCMenu control will choose an arbitrary item ID that is unique throughout the menu and all submenus. Likewise, items can be moved and copied between UCMenu toolbars. The resulting (new) menu items are assigned arbitrary IDs that do not conflict with any existing items.

The implication for the application is that it cannot depend on using item IDs to determine what

function was selected on a UCMenu toolbar. Instead, the application must use the action associated with the menu item to determine what function has been selected. A UCMenu control will send a `WM_CONTROL` message with a notification code of `UCN_ITEMSELECTED` when an item is selected. The application will not receive a `WM_COMMAND` message from a UCMenu control.

The application must associate the action string of the selected menu item with an application function. The toolkit sample program (`SAMP1.C`) uses a table that maps action strings to fixed ID values. The action of the selected item is looked up in the table and then a simple switch can be used to select the correct application function in a manner similar to the original `WM_COMMAND` processing in Figure 4. The ID values chosen for the table purposefully match the fixed item IDs used in the applications standard Presentation Manager menu bar (the application has both UCMenu toolbars and a standard Presentation Manager menu bar). Thus, the same switch can be used to process `WM_COMMAND` messages from the Presentation Manager menu

```
case WM_COMMAND:
    switch (SHORT1FROMMP(mp2)) {
        case CMDSRC_MENU:                /* Msg from a menu control */
            switch (SHORT1FROMMP(mp1)) {
                case ID_FILESAVE:          /* File->Save item selected */
                    // process file save
                    break;
                case ID_FILEOPEN:          /* File->Open item selected */
                    // process file open
                    break;
                ...
            }
        case CMDSRC_PUSHBUTTON:
            // process all pushbuttons
    }
    break;                                /* end of WM_COMMAND */
```

Figure 4. Processing a typical Presentation Manager menu selection.


```

UCMITEM *ItemData;
USHORT ItemID;
int i;

#define MAX_ACTIONS 4

struct { /* Table to map action strings to fixed IDs */
    USHORT CmdID; /* Unique ID for each menu item */
    PSZ Action; /* Unique 'action' string */
} ActionTable[MAX_ACTIONS] = {
    {ID_COPY, "Copy to Clipboard" },
    {ID_PASTE, "Paste from Clipboard" },
    {ID_LJUSTIFY, "Left Justify" },
    {ID_FONTS, "Select Fonts" },
};

case WM_COMMAND:
    if (SHORT1FROMMP(mp2) == CMDSRC_MENU) { /* From the PM menu */
        /* IDs of PM menu items are fixed ID_XXXX values */
        ItemID = SHORT1FROMMP(mp1); /* Extract item ID */
        WinSendMsg(hwnd, WM_USER+1, /* Send myself msg to do it */
            MPFROMSHORT(ItemID), 0L);
    }

    ...

case WM_CONTROL:
    if ( (SHORT1FROMMP(mp1) == ID_TOOLBAR) &&
        (SHORT2FROMMP(mp1) == UCN_ITEMSELECTED)) {
        if (ItemData->pszAction == NULL) /* Ignore if no action */
            return 0;
        /* Lookup action of this item and map to fixed ID_XXXX value */
        /* then send myself a message to process it. */
        for (i=0; i<MAX_ACTIONS; i++) {
            if (!strcmpi(ActionTable.Action[i], ItemData->pszAction)) {
                WinSendMsg(hwnd, WM_USER+1,
                    MPFROMSHORT(ActionTable.CmdID), 0L);
                return 0;
            }
        }
        WinMessageBox(...error, action not recognized...);
        return 0;
    }

    ...

case WM_USER+1: /* Process selections from any menu */
    switch (SHORT1FROMMP(mp1)) {
        case ID_COPY: /* do clipboard copy */
            break;
        case ID_PASTE: /* do clipboard paste */
            break;
        case ID_LJUSTIFY: /* do justification */
            break;
        case ID_FONTS: /* do font selection */
            break;
    }
    return 0;

```

Figure 5. Method to combine Presentation Manager and UCMenu selection processing.

and WM_CONTROL messages from the UCMenus. This reduces code size and keeps application-specific functions from being duplicated. The combined method for processing both Presentation Manager menu and UCMenu selections is shown in Figure 5.

A list of all UCMenu-related messages is shown in Figure 6. The messages are grouped by messages the UCMenu control sends to query the application for information (WM_CONTROL), messages sent to notify the application of significant events (WM_CONTROL), and messages sent by the application to the UCMenu to query or set various values (UCMENU_*).

UCMENUS DATA STRUCTURES

Two primary data structures are related to UCMenu controls. Associated with each UCMenu window is a set of data that describes various aspects of the overall toolbar appearance and behavior. This includes information such as style flags, menu colors, and text font. This is the UCMINFO structure shown in Figure 7. It is filled in by the application and used during creation of a UCMenu control.

Associated with each item of the menu are two linked data structures, as shown in Figure 8. The first is the standard Presentation Manager MENUITEM structure. That structure contains an "item handle" that is generally reserved for application use. A UCMenu uses the item handle to point to a UCMITEM structure, which provides UCMenu-specific item information. In effect, the MENUITEM structure is extended for UCMenu-specific data. It should be noted that any of the string pointers in the UCMITEM structure can be null.

These data structures are used extensively in the processing of messages to and from the UCMenu control.

Checkable Menu Items. The fact that a UCMenu can be extensively customized by the user has some important implications for applications with "checkable" style menu items. Like a Presentation Manager menu, any item in a UCMenu can be checked simply by setting the `MIA_CHECKED` attribute for that item. A menu item should be checkable if the action associated with it represents the toggling of some application state. A checked menu item is indicated in a UCMenu by drawing a thick border around it.

When a UCMenu item is created by the user, the `MIA_CHECKED` attribute is not set. Thus, if the action selected by the user for the new item is a toggle function, the application should update the current state of the item with the current state of the application.

Likewise, when a menu item is modified and assigned a new action, the current attributes are not changed. If the new action is a toggle function, the item attribute must be updated to reflect the current application state. If the action is not a toggle function, the `MIA_CHECKED` attribute should be cleared since the previous action may have been checked.

The UCMenu control will send a `WM_CONTROL` message with a notification code of `UCN_ADDEDITEM` or `UCN_ACTION` when a new item is added or an action is changed. When these messages are received, the application must examine the new action and set or clear the `MIA_CHECKED` attribute. The code in Figure 9 shows the processing of this message. Note that we have expanded the table used in Figure 5, which maps actions to fixed IDs, adding flags to determine if a particular action is supposed to be checkable and to track its check state.

When the application is notified of a selection on a checkable UCMenu item, it must toggle the

WM_CONTROL notification codes (UCMenu queries application for info)

<code>UCN_QRYTEMPLATEID</code>	Return resource ID of default menu template
<code>UCN_QRYTEMPLATEMODULE</code>	Return module handle of resource
<code>UCN_QRYRESBMP</code>	Return list of all bitmap resource IDs
<code>UCN_QRYACTIONLIST</code>	Send <code>UCMENU_INSERTACTION</code> for each supported action, then <code>UCMENU_ACTIONSINSERTED</code>

WM_CONTROL notification codes (UCMenu notifies application of events)

<code>UCN_ITEMSELECTED</code>	A UCMenu item was selected
<code>UCN_CMITEM</code>	A context menu item added by app selected
<code>UCN_SIZE,STYLE,FONT,COLOR</code>	A UCMenu style or attribute has changed
<code>UCN_ADDEDITEM</code>	A new item was added to the menu
<code>UCN_DELETEDITEM</code>	An item was deleted from the menu
<code>UCN_BITMAP</code>	The bitmap of an item was changed
<code>UCN_TEXT</code>	The text of an item was changed
<code>UCN_ACTION</code>	The action string of an item was changed
<code>UCN_HLP_*</code>	Help requested on customization dialog, occurs only if style <code>UCS_CUSTOMHLP</code>
<code>UCN_MOUSEMOVE</code>	The mouse moved on an item

UCMenu messages (from application to UCMenu control)

<code>UCMENU_ADDITEMSTOCM</code>	Add items to popup context menu
<code>UCMENU_INSERTACTION</code>	Response to <code>UCN_QRYACTIONLIST</code>
<code>UCMENU_ACTIONSINSERTED</code>	Signifies end of action list
<code>UCMENU_SETBGCOLOR</code>	Set menu and item background colors
<code>UCMENU_SETSTYLE</code>	Set menu style (<code>UCS_*</code> flags)
<code>UCMENU_UPDATE</code>	Update menu after style changes
<code>UCMENU_QUERYCOLOR</code>	Query menu bkgd and item bkgd colors
<code>UCMENU_QUERYFONT</code>	Query current menu font (text under bmps)
<code>UCMENU_QUERYFORCEDSIZE</code>	Query cx,cy if <code>UCS_FORCEDSIZE</code> style
<code>UCMENU_QUERYSIZE</code>	Query cx,cy for full menu display
<code>UCMENU_QUERYSTYLE</code>	Query menu style (<code>UCS_*</code> flags)
<code>UCMENU_QUERYUCMINFO</code>	Query complete menu create structure
<code>UCMENU_QUERYVERSION</code>	Query version number of UCMenus

Figure 6. Summary of UCMenu messages.

```
typedef struct {
    ULONG    cb;                // Size of this structure
    HMODULE  hModule;           // Module where the bitmaps are
    USHORT   NbOfCols;          // Number of columns in matrix style
    USHORT   NbOfRows;          // Number of rows in matrix style
    PSZ      pszBitmapSearchPath; // Path(s) to search for bitmap files
    PSZ      pszDefaultBitmap;   // Bitmap to use if bitmap specified for
                                // a menu item can't be loaded.
    ULONG    Style;              // Style of item (UCS_XXXX flags)
    ULONG    cx;                 // Fixed item size if UCS_FORCESIZE
    ULONG    cy;                 // Fixed item size if UCS_FORCESIZE
    LONG     BgBmp;              // RGB of bitmap color to be replaced
                                // with ItemBgColor.
    LONG     BgColor;            // RGB of UCMenu background (no items)
    LONG     ItemBgColor;        // RGB of UCMenu items background
    PSZ      pszFontNameSize;    // Font of the UCMenu or NULL for default
} UCMINFO, *PUCMINFO;
```

Figure 7. UCMINFO structure describing general characteristics of a UCMenu.

check state. In a standard Presentation Manager menu, it is sufficient to toggle the state of just the

menu item selected. Since a UCMenu may have the same (toggle) action on different items in the

same menu, the check state of all items with the toggle action must be updated.

This function has been incorporated into the UCMenu API set with the function `UCMenuSetActionAttr()`. This API will set the attributes of all items in the UCMenu with the specified action string. If the application has multiple UCMenu toolbars, all must be updated since the action may exist in any of them.

Customization Messages. Easy customization of the toolbar is one of the most useful and powerful features of this control. It gives the

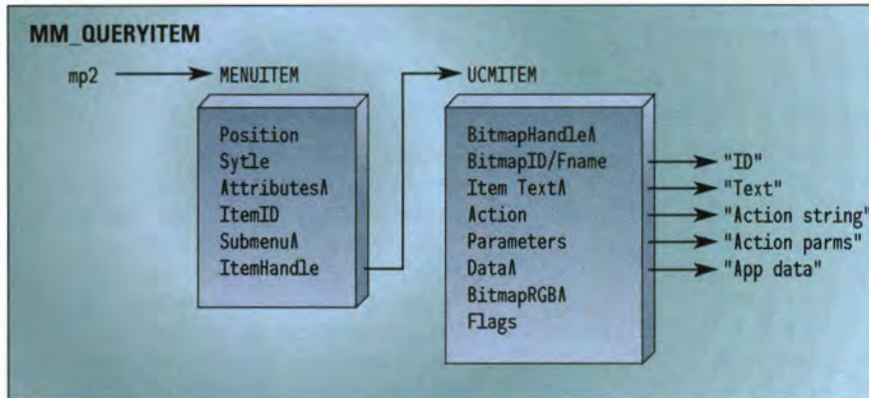


Figure 8. UCMenu item data structures.

Move menu items	Drag and drop menu items from one position to another. New position will be insertion point nearest the mouse cursor when the item is dropped. <code>MIS_SPACER</code> items can also be moved. Items may be moved from one toolbar to another.
Copy menu items	Drag and drop menu items with the Ctrl key pressed. Items may be copied from one toolbar to another.
Copy to submenu	Drag menu item with Alt key pressed and drop on another menu item. If the target item is not a submenu it will be changed to be one. The dropped item is inserted after the last item already in the submenu. Submenu items can be copied from one toolbar to another.
Delete menu items	Drag menu item to system shredder.
Create new items	Drag .BMP file type from desktop or drives folder onto a toolbar. A new item with the supplied bitmap will be created. The item will not have any text or action associated with it until provided with the context menu 'Edit item' option.
Render bitmap to file	A menu item that has file-supplied bitmap (the bitmap specification is a file name, not an application resource) can be dragged to the OS/2 desktop or a folder to create a .BMP file. The bitmap can also be dropped on the OS/2 Icon Editor desktop object to open and edit the source bitmap file.
Change colors	A color can be dragged from the system color palette onto the toolbar background or the item background. All items will have the same background color.
Change font	A font can be dragged from the system font palette onto the toolbar. The dropped font will be used for the text under the bitmaps.
Items can also be created, deleted, and edited with the context menu, which may be displayed by positioning the cursor on a menu item and pressing the right mouse button.	

Table 1. Direct manipulation capabilities of UCMenu toolbars.

user complete flexibility to add and delete menu items, rearrange the order, create submenus, and move and copy items between toolbars. Encapsulation of these functions into the UCMenu control means that there is very little application code required to support customization.

Although most of the customization function is encapsulated in the control itself, the application must respond to several messages for customization to work properly. The UCMenu control will send WM_CONTROL messages to its owner when it needs certain types of information to present on the customization notebook dialogues. The WM_CONTROL notification codes for customization are as follows:

- **UCN_QRYDEFAULTID:** This is sent when the user selects the *Load default* context menu item to restore the entire UCMenu to its default configuration. The application responds by supplying the resource ID of the menu template to be loaded.
- **UCN_QRYTEMPLATEMODULE:** This is also sent when the *Load default* context menu item is selected. The application must supply the handle of the module in which the UCN_QRYDEFAULTID resource is to be found.
- **UCN_QRYRESBMP:** This message is sent when the user wants to display a list of application-supplied bitmaps from which to choose. The application must allocate and construct an array of bitmap resource IDs.
- **UCN_QRYACTIONLIST:** This message is sent when the user has requested a list of actions from which to choose. The application responds by sending a series of UCMENU_INSERTACTION messages, each with a pointer to an action string and descriptive help text. After the last action has been inserted, the application must

```

MENUITEM MenuItem; /* Std PM menu item data */
UCMITEM *ItemData; /* UCMenu item data */
int i;
#define MAX_ACTIONS 4

struct { /* Table to map actions and track check states */
    USHORT CmdID; /* Unique ID for each menu item */
    PSZ Action; /* Unique 'action' string */
    BOOL Checkable; /* Is this item checkable */
    USHORT CheckStatus; /* Zero (not checked) or MIA_CHECKED */
} ActionTable[MAX_ACTIONS] = {
    {ID_COPY, "Copy to Clipboard", FALSE, 0},
    {ID_PASTE, "Paste from Clipboard", FALSE, 0},
    {ID_LJUSTIFY, "Left Justify", FALSE, 0},
    {ID_INSERT, "Toggle Insert Mode", TRUE, 0}, //<--Checkable
};

case WM_CONTROL:
    if ( (SHORT1FROMMP(mp1) == ID_TOOLBAR1) || /* Msg from any toolbar */
        (SHORT1FROMMP(mp1) == ID_TOOLBAR2))
        switch (SHORT1FROMMP(mp2)) { /* Notify code */
            UCN_ADDEDITEM: /* New item */
            UCN_ACTION: /* Chnged action */
                /* If an item is added or its Action changed check */
                /* if the selected action a checkable item. If so */
                /* we must set the current check state. */
                /* mp1=MenuID, Notify Code mp2=ItemID */
                if (!WinSendMsg (WinWindowFromID (hwnd, SHORT1FROMMP (mp1)),
                                MM_QUERYITEM,
                                MPFROM2SHORT (SHORT1FROMMP (mp2), TRUE),
                                MPFROM (MenuItem)))
                    return 0;
                /* Check that the UCMITEM->Action string pointers are OK */
                ItemData = (UCMITEM *) (MenuItem.hItem);
                if (ItemData==NULLHANDLE)
                    return 0;
                if (ItemData->pszAction==NULL)
                    return 0;
                /* Lookup action of this item in action table */
                for (i=0; i<MAX_ACTIONS; i++)
                    if (!strcmpi(ActionTable.Action[i], ItemData->pszAction)) {
                        /* If action is supposed to be checkable, set its */
                        /* check state to the current application state. */
                        if (ActionTable[i].Checkable) {
                            WinSendMsg(WinWindowFromID(hwnd, SHORT1FROMMP(mp1)),
                                MM_SETITEMATTR,
                                MPFROM2SHORT(SHORT1FROMMP(mp2), TRUE),
                                MPFROM2SHORT(MIA_CHECKED, ItemList[i].CheckStatus));
                        }
                        else { /* Action is not checkable, remove any check mark */
                            WinSendMsg(WinWindowFromID(hwnd, SHORT1FROMMP(mp1)),
                                MM_SETITEMATTR,
                                MPFROM2SHORT(SHORT1FROMMP(mp2), TRUE),
                                MPFROM2SHORT(MIA_CHECKED, "MIA_CHECKED"));
                        } /* if checkable */
                    } /* if action found in table */
                } /* switch on notify code */

```

Figure 9. Setting "checked" attribute on added and modified UCMenu items.



send UCMENU_ACTIONSINSERTED to indicate the list is complete.

- UCN_HLP_*: This is a set of messages sent when the UCS_CUSTOMHLP menu style is used and the user requests help on a customization dialogue. By default, the UCMenu control will supply the needed help text.

The handling of all these messages is demonstrated in the SAMP2.C sample program.

CONTROL OF CUSTOMIZATION

The highly flexible customization built in to the UCMenu control may not be desirable for all applications or situations. For example, the ability to move items from one menu to another within an application may not make semantic sense. An application with multiple windows performing different functions may not want to allow the user to drag items from the

toolbar in one window to the toolbar in another. The result might be actions that make no sense in the context of the window. The application may also want to provide its own customization methods to replace or supplement those of UCMenus.

By default, all the direct manipulations shown in Table 1 and all the context menu options shown in Figure 2 are allowed. Various style flags (UCS_*) can be used by the application to control which direct manipulation and context menu functions are supported for that instance of the menu. At the extreme, the application can set a style of UCS_STATIC to prevent any customization at all. This style will disable all direct manipulation functions and the context menu. Other style flags allow selective disabling of specific customization functions, such as disabling drag-and-drop between menus (but allowing it within a menu), disabling drops to the system shredder, removing the "Edit item..." context menu option, and so on. Table 2 summarizes the customization style flags.

An application can also supply its own customization functions via the UCMenu context menu. By sending the control a UCMENU_ADDITEMSTOCM message, the application can add its own options to the context menu. The UCMenu control will send a WM_CONTROL message with a notify code of UCN_CMITEM when such an option is selected. A sample of adding an application customization function to the context menu is shown in Figure 10.

CREATING A UCMENU CONTROL

Conceptually, creating a UCMenu control is the same as creating a Presentation Manager menu control. The menu structure and items are defined in textual form in a resource file (.RC) using MENU, MENUITEM, and SUBMENU statements. The re-

Moving Menu Items

UCS_NO_DM_M_TO_OTHER	Disable drag-move of items to another UCMenu
UCS_NO_DM_M_FROM_OTHER	Disable drop-move of items from another UCMenu
UCS_NO_DM_M_FROM_INSIDE	Disable drag-drop move (rearrangement) of items inside this UCMenu

Copying Menu Items (CTRL key drag/drop)

UCS_NO_DM_C_TO_OTHER	Disable drag-copy of items to another UCMenu
UCS_NO_DM_C_FROM_OTHER	Disable drop-copy of items from another UCMenu
UCS_NO_DM_C_FROM_INSIDE	Disable drag-drop move (rearrangement) of items inside this UCMenu
UCS_NO_DM_ALT	Disable use of ALT key to copy items to submenus

Desktop Interaction

UCS_NO_DM_DISCARD	Disable drag of items to system shredder to delete them
UCS_NO_DM_RENDER_TO_BMP	Disable drag of items to desktop to create .BMP files
UCS_NO_DM_RENDER_FROM_BMP	Disable drop of .BMP files to create new items

Context Menu Control

UCS_NO_CM_MENU_STYLE	Remove "Change style..." option from context menu
UCS_NO_CM_MENU_IMPORT	Remove "Import..." option from context menu
UCS_NO_CM_MENU_EXPORT	Remove "Export..." option from context menu
UCS_NO_CM_MENU_DEFAULT	Remove "Load default" option from context menu
UCS_NO_CM_ITEM_CREATE	Remove "Create item..." option from context menu
UCS_NO_CM_ITEM_DELETE	Remove "Delete item" option from context menu
UCS_NO_CM_ITEM_EDIT	Remove "Edit item..." option from context menu

Combination Styles

UCS_NO_DM	Disable all direct manipulation
UCS_NO_CM	Disable context menu
UCS_STATIC	Disable all direct manipulation and context menu

Table 2. UCMenu customization style flags.

source file is compiled with the resource compiler into binary resource format and appended to the application executable (or a DLL). The application calls an API to read the binary resource and create the visual Presentation Manager windows and underlying data structures. Finally, the window is placed and sized on a parent window.

Resource Files. The textual resource file specifies the complete structure of the menu and the contents of the menu items. However, the `MENUITEM` resource statement does not have the syntactic flexibility to allow complete specification of all the data associated with a UCMenu item. Therefore, the text field of the `MENUITEM` statement is interpreted to contain four extra fields of information not found on standard Presentation Manager menu items: bitmap specification, action string, parameter string, and a data string. Each of these fields is separated by a single character defined as the first character of the string. A typical resource file for a UCMenu is shown in Figure 11.

Some notes on UCMenu resource files:

- The menu item IDs specified in the resource file are arbitrary. Sequential numbering is used to help keep menu items unique but is not necessary.
- A new menu item style `MIS_SPACER` is defined for UCMenus. It produces a blank space in the menu to separate menu items into groups.
- The 'Data' field is not seen by the user and is for application use. Note that new items created by customization will have no 'data' value.

The UCMINFO Structure. To complete the process of creating a UCMenu, the application must construct a `UCMINFO` data structure (as shown in Figure 7) and fill in all the relevant

```
CMITEMS CMItem;    /* Describes one context menu item */
static hwndUCM;    /* Handle of UCMenu window          */

case WM_INITDLG: (or WM_CREATE):

    //...after UCMenu is created and handle is in hwndUCM...

    /* Add items to UCM context menu (we just have one) */
    CMItem.ID = ID_MYCUSTOMIZATION;
    CMItem.pszItemText = "Reset style";
    WinSendMsg(hwndUCM, UCMENU_ADDITEMSTOCM,
                MPFROMLONG(1L), MPFROMP(&CMItem));

    ...

case WM_CONTROL:
    if ((SHORT1FROMMP(mp1)==ID_TOOLBAR) &&    // From the toolbar menu
        (SHORT2FROMMP(mp1)==UCM_CMITEM)) {    // Our context option
        if (SHORT1FROMMP(mp1) == ID_MYCUSTOMIZATION) {
            /* Reset style of the menu to default. To be complete, */
            /* this should also reset colors and fonts in case they */
            /* were changed via drop from system palettes.          */
            WinSendMsg(hwndUCM, UCMENU_SETSTYLE,
                MPFROMLONG(UCS_FRAMED|UCS_CHNGBMP|CUSTOMHLP),
                MPFROM2SHORT(0,0));
        }
    }
}
```

Figure 10. Adding an option to the UCMenu context menu.

fields. The RGB values of the structure are used for color mapping the bitmap background pixels. UCMenus supplies a means for bitmap background pixels to be automatically mapped to the current UCMenu background color. The `BgBmp` value of `UCMINFO` tells UCMenus what background color was used in the design of the bitmaps. That color will be replaced with the `ItemBgColor` before the bitmaps are displayed wherever that color appears in the bitmap. The `ItemBgColor` should be set to the `SYSCLR_MENU` color returned by `WinQuerySysColor()`. A different color may be specified in the `BgColor` field. This will be the color used wherever there are no items on the toolbar.

Once the `UCMINFO` structure is filled in, the function `UCMenuCreateFromTemplate()` or `UCMenuCreateFromResource()` is called. The template version is used if the application has manually loaded a

previously saved UCMenu template. The resource version will create a UCMenu from a menu template in the application's resources. The parameters are shown in Figure 12. The creation functions return two window handles. The function result is the window handle of interest for the application; the other may be ignored.

PLACEMENT AND SIZING OF A UCMENU

Toolbars are generally placed along one edge of the application window frame. Toolbars may be attached to the top, bottom, left, or right edge of the window. Figure 1 shows a frame window with four toolbars, one on each edge of the frame.

It is possible to simply place and size the toolbar into the client window of a standard Presentation Manager window. However, this simple approach introduces a lot of complexity when the application



```

BITMAP 6 "new.bmp"
BITMAP 1 "open.bmp"
BITMAP 2 "save.bmp"
BITMAP 62 "styles.bmp"
BITMAP 63 "bold.bmp"
BITMAP 64 "italic.bmp"
BITMAP 8 "help.bmp"

/* Text strings are interpreted as: */
/* */
/* "<c>Text<c>BitmapID<c>ActionStr<c>ParamStr<c>DataStr" */
/* */
/* where <c> is any character that does not appear in the strings. */
/* */
/* Item style MIS_SPACER produces a gap in the menu bar. */
/* */

MENU ID_COMMANDBAR LOADONCALL MOVEABLE DISCARDABLE
BEGIN
    MENUITEM "/New/6/New",          1, MIS_TEXT
    MENUITEM "/Open/1/Open File",   2, MIS_TEXT
    MENUITEM "/Save/2/Save File",   3, MIS_TEXT
    MENUITEM "",                    4, MIS_SPACER
    SUBMENU "/Styles/62/Styles Submenu", 5, MIS_TEXT
    BEGIN
        /* Use a different delim here so we can use "/" in the strings */
        MENUITEM "!Bold!63!Bold/Style", 6, MIS_TEXT | MIS_CHECKABLE
        MENUITEM "!Italic!64!Italic/Style", 7, MIS_TEXT | MIS_CHECKABLE
    END
    MENUITEM "",                    8, MIS_SPACER
    MENUITEM "/Help/8/Show Help",    9, MIS_TEXT
END

```

Figure 11. Resource file statements for a UCMenu.

must properly scale and paint the client window. If the frame window is a dialogue, proper size and placement is even more complex since the application does not paint and there is no separate client window.

The proper solution to toolbar placement is to make the toolbar a *frame control*. A frame control is a child of the frame window and participates in a special message protocol. Standard frame controls include the title bar, scroll bars, Presentation Manager menu bar, min/max buttons, and the system menu. The frame controls move and size with the frame window and keep proper position in the frame through the frame control messages.

Making a window into a frame control requires a thorough knowl-

edge of how Presentation Manager frame windows and message protocols work. Adding a frame control to a dialogue window is even more complex since it may require the movement of all the dialogue controls. All the functions necessary to make UCMenu windows into frame controls are supplied in a set of utility routines in the toolkit. These routines are not a formal part of the UCMenu control but allow an application to easily place UCMenu controls along any edge of a frame or dialogue window. By using these utilities, an application can be completely removed from any sizing or positioning considerations for the toolbars.

The frame control utilities consist of two functions: one to add a UCMenu to a frame or dialogue

window, the other to remove it. The UCMutils() functions are shown in Figure 13. The utilities support up to four toolbars per frame or dialogue window (one each at the top, bottom, left, and right edges). By calling the add/remove frame control utilities, a toolbar can be toggled on and off.

The UCMutils() functions encapsulate all the complexity of frame control management, freeing the application from that task. With just a simple function call, the toolbar can be positioned in the frame and will maintain its proper position as the frame is sized and moved. Source code for the utilities is provided in the toolkit.

A DYNAMIC STATUS BAR

One drawback to graphical menu bars is the cryptic nature of the information presented. Even the most creative bitmaps can fail to convey the meaning of a function. Text under the bitmap, as allowed by UCMenus, helps the user quickly understand the functions on the toolbar. However, even that may not be enough for the casual user to grasp the meaning of a graphic menu item.

A dynamic status bar can be implemented by the application to automatically show a description of each menu item as the mouse pointer is moved on the toolbar. To help the application implement this feature, a UCMenu with the UCS_PROMPTING style will send WM_CONTROL messages with a notify code of UCN_MOUSEMOVE when the pointer moves on the menu bar. The message supplies the menu item ID of the item under the mouse pointer. Using this item ID, the application can obtain the item data structures and determine the action string associated with that item. The action string may be displayed directly, or it may be used to look up a more complete description for display to the user. The SAMP2.C

sample program uses the lookup method to display a longer description of the menu items action.

SAVING/RESTORING UCMENUS

When an application with customizable menus is closed and restarted, the user expects the menus to resume their customized state. The application must save the complete state of the UCMenus when closing, including the menu attributes in the UCMINFO structure and the actual menu item structure (order of items, text, bitmaps, action strings, and so on).

The UCMenus API provides several functions to aid the application in saving and restoring the customized state of UCMenu toolbars. The function `UCMenuMakeTemplate()` will create a binary in-storage representation of the complete UCMenu item structure. The binary image will include all text, bitmaps, action strings, and arrangements of menu items. The corresponding APIs, `UCMenuCreateFromTemplate()` and `UCMenuNew`, create and replace UCMenu controls from an in-storage binary template.

Figure 14 shows the processing of `WM_SAVEAPPLICATION` for the SAMP2.C sample program. This sample has two UCMenu toolbars, but for brevity, the code for only one is shown. The UCMenu style, colors, and font, as well as the binary templates, are stored in the application .INI file. During initialization, the styles, colors, font, and templates are read from the .INI file and are used to recreate the customized toolbars.

SUMMARY

The UCMenu control provides a powerful, slick-looking, and easy-to-use toolbar control for OS/2 Presentation Manager applications. The encapsulation of extensive customization capabilities gives application writers a powerful, customizable toolbar with min-

imal code. For the user, UCMenus provides an easy-to-use toolbar that is simple to customize with intuitive drag-and-drop and context menus.

Examination of the SAMP2.C sample program shows how a typ-

ical application might be structured with respect to menus and menu selection processing. The sample shows the usage of most of the UCMenu-specific messages and fully implements all features of the UCMenu control.



```

HWND UCMenuCreateFromResource( // Returns handle of new toolbar
    HAB      hab,           // PM anchor block of application
    HWND     hParent,       // Parent window
    HWND     hOwner,        // Owner window (recv messages)
    ULONG    ulStyle,       // CMS_VERT,HORZ,MATRIX plus WS_* flags
    LONG     x, LONG y,     // Position
    LONG     cx, LONG cy,   // Size
    HWND     hInsertBehind, // Z order
    ULONG    ulID,          // Window ID new toolbar is to have
    HMODULE   hmodResource, // Resource module (NULLHANDLE for EXE)
    USHORT    usMenuID,     // ID of menu in resources
    UCMINFO   *pUCMInfo,    // Ptr to UCMINFO data
    HWND     *phTextMenu)   // (returned) PM menu handle (don't use)

HWND UCMenuCreateFromTemplate( // Returns handle of new toolbar
    HAB      hab,           // PM anchor block of application
    HWND     hParent,       // Parent window
    HWND     hOwner,        // Owner window (recv messages)
    ULONG    ulStyle,       // CMS_VERT,HORZ,MATRIX plus WS_* flags
    LONG     x, LONG y,     // Position
    LONG     cx, LONG cy,   // Size
    HWND     hInsertBehind, // Z order
    ULONG    ulID,          // Window ID new toolbar is to have
    VOID     *Template,      // Ptr to instorage template
    UCMINFO   *pUCMInfo,    // Ptr to UCMINFO data
    HWND     *phTextMenu)   // (returned) PM menu handle (don't use)

BOOL UCMenuNew(             // Update toolbar with new template
    HWND     hwndUCMenu,     // Toolbar window
    VOID     *Template)      // Instorage template

VOID *UCMenuMakeTemplate(   // Make instorage template from toolbar
    HWND     hwndUCMenu,     // Toolbar window handle
    LONG     *TemplateSize)  // (return) Size of template

```

Figure 12. Selected UCMenu APIs.

```

UCMUtilsAddToFrame(HWND hwndFrame,           // Frame or dialog window
                   HWND hwndUCMenu,         // UCMenu to add to frame
                   ULONG Placement)          // Where to place toolbar:
                                           // UCMUTILS_PLACE_TOP
                                           // UCMUTILS_PLACE_BOTTOM
                                           // UCMUTILS_PLACE_LEFT
                                           // UCMUTILS_PLACE_RIGHT

UCMUtilsRemoveFromFrame(HWND hwndFrame,      // Frame to remove toolbar
                        HWND hwndUCMenu)     // UCMenu to be removed

```

Figure 13. Frame control management utility functions.


```

/* Structure to easily save/restore menu information in INI file */
typedef struct {
    ULONG CmdStyle;        /* Command menu style bits */
    ULONG CmdCx;           /* Forced size (x) */
    ULONG CmdCy;           /* Forced size (y) */
    ULONG CmdSize;         /* Size of menu template */
    ULONG CmdBgColor;      /* Toolbar background */
    ULONG CmdItemBgColor;  /* Item background */
    char CmdFont[32];      /* Font name */
} IniDataStruct;

BOOL ProfileFound = FALSE; /* INI file existed */
UCMINFO UCMInit;          /* UCMenu creation data */
PVOID CmdTemplate;        /* Ptr to menu template */
HWND CommandBar;          /* UCMenu window handle */
...

WM_CREATE: // (or WM_INITDLG, if a dialog):

/* If we have saved a previous UC menu configuration in the .INI */
/* file, restore its contents and style information. */

IniHandle = PrfOpenProfile(Hab, "myapp.ini");
if (IniHandle != NULLHANDLE) {
    /* If one value is here, they all should be */
    ULValue = sizeof(IniDataStruct);
    if (PrfQueryProfileData(IniHandle, "APPNAME",
        "IniData", &IniData, &ULValue)) {
        CmdTemplate = malloc(IniData.CmdSize);
        PrfQueryProfileData(IniHandle, "APPNAME", "CmdTemplate",
            CmdTemplate, &(IniData.CmdSize));
        ProfileFound = TRUE;
    }
    PrfCloseProfile(IniHandle);
}

// Initialize all fields of the UCMInit structure here...then:
if (ProfileFound) { /* Create menu from saved template */
    UCMInit.Style = IniData.CmdStyle; /* Restore menu style */
    UCMInit.cx = IniData.CmdCx; /* Restore forced size */
    UCMInit.cy = IniData.CmdCy; /* Restore forced size */
    UCMInit.BgColor = IniData.CmdBgColor; /* Colors */
    UCMInit.ItemBgColor = IniData.CmdItemBgColor;
    UCMInit.pszFontNameSize = IniData.CmdFont; /* Font */
    CommandBar = UCMenuCreateFromTemplate(...CmdTemplate...);
    free(CmdTemplate);
}
else { /* Create menu from resource file */
    CommandBar = UCMenuCreateFromResource(...ID_TOOLBAR...);
}
...

WM_SAVEAPPLICATION:

/* Create incore template representation of the menu */
CmdTemplate = UCMenuMakeTemplate(CommandBar, &(IniData.CmdSize));
/* Open INI file and save template and style info */
IniHandle = PrfOpenProfile(Hab, "myapp.ini");
if (IniHandle != NULLHANDLE) {

```

Figure 14. Saving and restoring a customized toolbar in the application .INI file (continued on page 63).


```

PrfWriteProfileData(IniHandle, "APPNAME", "CmdTemplate",
                    CmdTemplate, IniData.CmdSize);
/* We must also save the style bits, size, colors, fonts. */
/* Colors and fonts can be modified via system palettes. */
WinSendMsg(CommandBar, UCMENU_QUERYUCMINFO, MPFROMP(&UCMInit), 0L);
IniData.CmdStyle      = UCMInit.Style;
IniData.CmdCx         = UCMInit.cx;
IniData.CmdCy         = UCMInit.cy;
IniData.CmdBgColor    = UCMInit.BgColor;
IniData.CmdItemBgColor = UCMInit.ItemBgColor;
if (UCMInit.pszFontNameSize != NULL)
    strcpy(IniData.CmdFont, UCMInit.pszFontNameSize);
else (IniData.CmdFont)[0] = '\0';
UCMUtilsFreeUCMInfoStrings(&UCMInit);
PrfWriteProfileData(IniHandle, "APPNAME", "IniData", &IniData,
                    sizeof(IniDataStruct));
PrfCloseProfile(IniHandle);
}
UCMenuFree(CmdTemplate);
UCMenuFree(ClrTemplate);
break;

```

Figure 14. Saving and restoring a customized toolbar in the application .INI file (continued from page 62).

Mark McMillan is an advisory engineer with IBM's Integrated Systems Solutions Corp. (ISSC) subsidiary at Research Triangle Park, N.C. Mark joined IBM in 1983 working on electronic design automation software. He received his B.S. at the University of Michigan and completed his master's degree in computer engineering at Duke University. He was responsible for the concept, design, and implementation of the Communications Manager Mouse Support (CMMouse) product for OS/2, DOS, Windows, and AIX. He is also the lead developer on an IBM client/server directory system.

Gennaro Cuomo is senior manager of the Groupware technology department at the IBM T.J. Watson Research Center. Gennaro did his graduate studies in computer science at Syracuse University and N.Y. Polytechnic University. In 1981, he joined IBM Research's Physical Science department, where he worked on lab automation projects for five years as a Coop and Graduate work study student. During the past eight years, Gennaro has contributed to the design and development of

the following IBM products: OS/2 Enhanced Editor, OS/2 TCP/IP, and UltiMail. Currently, Gennaro is active in forging IBM's new direction in application development and groupware.

Dr. Jürg von Känel is manager of the Multimedia Messaging department at the IBM T.J. Watson Research Center. He joined IBM in 1985 in Switzerland and has worked in application development and research. In 1991, he moved to the research

lab in Yorktown where he took on the development of multimedia messaging systems. Currently, he is the chief architect of IBM's UltiMail multimedia mail system.

Guillaume Le Stum is a programmer in the Compound Documents Technology group at IBM Research, Hawthorne, N.Y. He has worked on the EPM editor, the UCMenus toolbar, and Ultimail since 1993, when he finished his studies at Ecole Centrale in Paris, France.

REFERENCES

Code and Samples

The UCMenus custom control, frame control utilities, and fully functional sample programs are available on CompuServe's OS2DF2 Forum, OS/2 Developer section. Also included is a complete online (INF format) UCMenus programming reference. Download file UCMENU.ZIP.

Publications

OS/2 2.0 Technical Library Presentation Manager Programming Reference II (IBM S10G-6265).

OS/2 2.0 Technical Library Presentation Manager Programming Reference III (IBM S10G-6272).

A Perfect Fit Win-OS/2 Support

with the
Twain for OS/2 Toolkit



For more information
or
to order your developers kit call:

ST Solution Technology, Inc.
1101 S. Rogers Circle Bldg. 14
Boca Raton, FL 33487
(407) 241-3210 fax (407) 997-6518

Circle Reader Service Number 21

ASIAN OS/2 DEVELOPER SUPPORT



Get help porting OS/2 applications for
Japanese, Chinese, or Korean.

Experienced guidance from the leading
double byte character set OS/2 developer.
Training or consulting on either an hourly or
long term basis.



MicroBurst, Inc.

9035 Shady Grove Court
Gaithersburg, MD 20877

(301) 330-2995

Fax: (301) 330-8609

CompuServe: 70334,3616

Internet: 70334.3616 @
COMPUSERVE.COM.

Circle Reader Service Number 22

ADVERTISER INDEX

Reader Service	Company Name	Page
8	Artisoft.....	23
16	Athena Design.....	43
35	Compuware.....	17
20	Development Technologies.....	47
25	Essex Systems.....	72
9	Hockware.....	25
—	IBM.....	11
6	IBM.....	13
24	Indelible Blue.....	72
34	Inmark.....	COV4
11	JBA.....	29
27	Jagre.....	72
31	MHR.....	80
5	Metaware.....	21
22	Micro Burst.....	64
32	MicroEdge.....	80
3	Micro Focus.....	2
17	Mitnor Software.....	44
10	Object People.....	27
18	On-Line Data.....	45
29	Opt-Tech.....	72
26	Perez Computing Services.....	72
19	Pinnacle.....	46
4	Programmer's Paradise.....	4
23	Quadron.....	69
33	Rimstar.....	COV3
12	Softbridge.....	38
13	Solution Technology.....	38
21	Solution Technology.....	64
30	Solution Technology.....	72
2	Tech Bridge.....	1
14	Van Nostrand Reinhold.....	39
1	Veritas.....	COV2
15	Watcom.....	41



This article provides a brief overview of things to keep in mind when developing applications for OS/2.

By DIETER BABUTZKA

An OS/2 Performance Primer

During the development of a new application, performance isn't always given the attention it deserves, particularly in the early phases of a project. This article reviews some basic guidelines that should help improve your OS/2 application's performance.

PROCESSES AND THREADS

A large number of processes and threads in your application generate overhead from process and thread switches. It makes sense to use multiple processes and threads only when they can run independently or are used for protection reasons. Process switches consume more time than thread switches. Use multiple threads only if the application consists of parts that are able to run concurrently. Multiple processes should be used to prevent accidental overwriting of memory and to be completely independent of other parts of the application (crash protection).

FUNCTION CALLS

Although it may be fashionable to create a lot of very small functions, this can lead to performance problems since there is a significant overhead for function calling. It is important to avoid a high number of calls from 16-bit to 32-

bit applications because this leads to a perceptible overhead.

HOST OR LAN CALLS

Most newly developed applications are not stand-alone but work in a client/server environment. It is important to keep the number of interplatform calls to a minimum because every communication step generates an overhead that leads to longer response times.

MEMORY CONSUMPTION

In a modern OS/2 environment, there is almost unlimited virtual storage available for every application. But using a large amount of storage reduces not only the speed of your application but also the storage available for other applications, so their speed is also reduced. Try running the code example in Figure 1 while running another application. You'll notice that the whole system is much less responsive. Be careful! This program will use up to 40 MB of memory, and eventually your swap drive will be full.

DYNAMIC LINK LIBRARIES

The use of DLLs makes a lot of sense because the storage used for the whole system can be reduced. To reduce loading times, the number of DLLs should not be too large. We were able to reduce



startup time more than 10% after we reduced the number of DLLs from 20 to only one.

DATA FILTERS AND DATA CONVERSION

Sometimes much more data is transferred from a server to a client

than is really used. This should be avoided by using data filters at the first possible point, usually on the server. Often, the data is converted many times before it reaches the user. Try to convert data only once. Following is a typical example of the usefulness of a data filter. Let's

say you work on an OS/2 machine with a host database as a server and want to get data from this database. The database should consist of a large number of database objects. If you want to get all objects where the object name starts with an A, you'll have mainly two options:

- Get all objects from the host and look for all objects where the object name starts with an A.
- Pass a filter to the host and capture only these objects from the database where the object name starts with an A.

Normally, the data has to be converted in the object format defined on the workstation. Try to keep this conversion as easy as possible—meaning the organization of the objects on the workstation should not differ too much from the implementation on the host.

APPLICATION BUILDERS

An application builder makes sense in the development of large applications. For a small project, an application builder may not be advisable because it generates too much overhead.

During the coding process, it is important to ensure the performance of an application (or of selected key sections) by using test scenarios. These scenarios should be defined during the design phase, and if possible, goals should also be specified. Attention should be paid to mainly two items.

Path Length. Every developer should use a software performance analysis utility to measure the run time and the active time of a component, such as EXTRA. EXTRA is an abbreviation for the IBM C/C++ Execution Trace Analyzer. EXTRA is part of the IBM C Set++ compiler, but other performance analysis applications are also available for many other compilers. With EXTRA, you can analyze the

```
/******  
/* SPMTEST2 (program for SPM/2) */  
/* Just consumes RAM */  
/* It allocates every 0.5 sec. 40kB of memory, up to 40 MB */  
/* */  
/* COPYRIGHT: */  
/* ----- */  
/* Dieter Babutzka Boeblingen Software Systems 03.08.93 */  
/* */  
/******  
/* NOTE: The program also requires the IBM TOOLKIT 2.0 headers */  
/* */  
/******  
  
#define INCL_DOS  
#define INCL_DOSMEMMGR  
#include <os2.h>  
#include <malloc.h>  
#include <bsememf.h>  
#include <process.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
int main( int argc, char *argv' )  
{  
    PVOID memadd=NULL;  
    ULONG size=40000;  
    PCHAR help=NULL;  
    int i=0;  
    /* Do nothing, but just consuming memory */  
    i=0;  
    for (i=1;i<1000;i++) {  
        DosAllocMem(&memadd,size,f  
        ALLOC); help =  
        (PCHAR)memadd;  
        memset(memadd,'\0',size);  
        strcpy(help,"I'm a memory eater!");  
        DosSleep(500);  
    } /* endfor */  
    return;  
}
```

Figure 1. Code example for an OS/2 application allocating a large amount of memory.

run time, the active time, the number of calls, and who is called by whom.

Figure 2 shows the statistical output of EXTRA for a test program. You'll get information about the active modules (EXEs and DLLs) and a lot of details about the function called during the test scenario.

This is very important in complex systems coded in object-oriented programming languages. In this environment, it may not be known which calls occur and how often the various methods are called.

Memory Consumption. Although in the OS/2 concept a lot of virtual

memory is available for every application, you have to keep in mind that only a limited amount of real storage is available. We use the IBM SPM/2 2.0 product to check memory allocation.

With THESEUS/2, which is part of SPM/2, you can determine the memory used in various allocations (Decommitted, Committed, Accessed, Present, Swapped out). It can also be determined when an application allocates memory and does not free it again. This is not a real problem if the process to which this program belongs terminates. But if the program works as a server, which is designed to run continuously, it is very important

to look for memory that needs to be freed. This can slow down performance dramatically.

Figure 3 gives a short view about the actual memory division of all running programs. This is not the working set of the program but primarily a snapshot of the memory.

Although EXTRA helps in finding application bottlenecks, it is important to measure performance test scenarios with accurate precision. To do this, it is important to measure with as little overhead as possible. EXTRA and other software tools normally operate on debugging and nonoptimized code, so loading and startup times



Statistics - E:\XTRA\GOS.TRC					
File Options Help					
E:\XTRA\GOS.EXE run 10/28/94 at 04:10:01 PM					
36% E:\XTRA\GOS.EXE					
64% E:\XTRA\BUSGOSAI.DLL					
Executables generating events: 2 of 27					
Functions generating events: 74					
Function Names	% Run	% Active	Total Calls	Run Time	Active Time
main	29	100	1	2,080	7,080
ResCreateBitmapOfPS.BUSGOSAI	26	26	1	1,870	1,870
FntSetAttr.BUSGOSAI	10	10	15	702	704
ResCreateMemHPS.BUSGOSAI	8	8	3	545	545
GosMainCreateViewportWindow	5	5	1	320	338
GosInitLogFiles.BUSGOSAI	4	4	1	277	277
GosInit.BUSGOSAI	3	55	1	218	3,901
LogString.BUSGOSAI	2	2	90	124	124
FraOpenBitmapFile.BUSGOSAI	1	1	6	95.0	95.0
GosInitGlobal.BUSGOSAI	1	47	1	88.5	3,294
LogMemoru.BUSGOSAI	1	1	132	88.3	88.3

Figure 2. Statistical output of EXTRA for a test program.



are higher than in production code.

The tool we used to make this measurement accurately while generating little overhead was an IBM internal hardware tool. It is possible for IBM customers to borrow this hardware to analyze applications. Performance analysis is also available as an IBM service offering, which is available through the author.

When using this tool, hooks are written to a special card and transferred to another machine for analysis after the test scenario has finished.

Through the help of a software tool, hooks are included automatically at the beginning and the end of each function. After the execution of the test scenario, information is available for each function, consisting of:

- A trace listing showing when each function was called, who called it, and how long it was active
- Statistics about the active and idle periods (for example, waiting for an event, host communications, and so on)
- Statistics about each observed function, how often it was

called, time spent in each function, and the overall run time.

With this data, it is not difficult to find the bottlenecks of an application, although they still have to be fixed.

Some additional considerations should be kept in mind:

- Even a hardware test tool has some overhead—usually around 2%. But this overhead is subtracted from its measurement, so its accuracy is very high. After the overhead is subtracted, the difference between instrumented and noninstrumented measurements is less than 1%.

RAM Usage by Process							
Functions Output Mark/Find Misc Help							F1=Help
RAM Usage by Process:							
private			owned shared				
bytes	Kbytes	Mbytes	bytes	Kbytes	Mbytes	who	
00877000	8668	8.465	00219000	2148	2.098	system	
00000000	0	0.000				sysinit	
00000000	0	0.000	00002000	8	0.008	LANMSGEX	
00000000	0	0.000				LANDLL	
00006000	24	0.023				CNTRL	
0003A000	232	0.227	000E6000	920	0.898	PMSHL32	
00000000	0	0.000				HARDERR	
00000000	0	0.000	00010000	64	0.063	STOPLAN	
00064000	400	0.391	00037000	220	0.215	PMSHL32	
00000000	0	0.000				CMD	
0001A000	104	0.102	00006000	24	0.023	PMDIARY	
0000A000	40	0.039	00001000	4	0.004	TIMEEXEC	
00000000	0	0.000	00006000	24	0.023	CMSTART	
00010000	64	0.063	00040000	256	0.250	CMD	
00000000	0	0.000				EPWROUT	
00000000	0	0.000				EPWMUX	
00000000	0	0.000				ACSRASP	
00001000	4	0.004	00013000	76	0.074	ASP000	
0000D000	52	0.051	00005000	20	0.020	CMKFMSMI	
00000000	0	0.000				MUGLRQST	
00040000	256	0.250	0009D000	628	0.613	ACS3EINI	
00010000	64	0.063	0000C000	48	0.047	WKSTA	
00000000	0	0.000				WKSTAHL	
00000000	0	0.000				LSCLIENT	
0000E000	56	0.055	00008000	32	0.031	MSRV	
00005000	20	0.020	00008000	32	0.031	NETPOPUP	
00000000	0	0.000				CMD	
00000000	0	0.000				LSDAEMON	
00005000	20	0.020				VDM	
00057000	348	0.340	00003000	12	0.012	E	
0003B000	236	0.230	00018000	108	0.105	CMD	
00050000	320	0.313	0000D000	52	0.051	THESEUS2	
00AA7000	10908	10.652	00491000	4676	4.566	total RAM in use	
00067000	412	0.402	free RAM				
00F9F000	15996	15.621	total of all RAM pages found (Pvt + Shr + Free)				
< End of THESEUS2 (v 2.0.1c) output @ 17:05:58 on 10-27-1994 >							

Figure 3. Snapshot of the memory.

- No programmer likes to put hooks into his code for performance measurements. Therefore, an automatic instrumentation program is available that puts the hooks into the code without any manual interaction. These hooks are then written to the hardware for later analysis.

The conclusion of our experience is that it is important to look at the performance during the design phase because it is hard to improve performance when coding is nearly complete.

Dieter Babutzka has been with IBM since 1989 and works as a Senior Associate in the Boeblingen Programming Laboratory. He has worked in several MVS programming, planning, and testing groups for products like HCD and RMF. On the workstation side, he was responsible for the performance analysis of several OS/2 products. He received a diploma in physics from the Eberhard-Karls-University in Tuebingen in 1986. Dieter can be reached at dbabutzka@vnet.ibm.com.

REFERENCES

Program	Part Number
IBM C Set ++ for OS/2 Version 2.1	82G3732
IBM System Performance Monitor/2 2.0	96F8379

Quadron tools cut 32% off the average development time.

Take it from our customers. A recent survey determined that, by using Quadron communications development tools, they cut 32% off their estimated development time.

Increase your efficiency

Our development software allows you to shift communications tasks from the CPU on a PC to a specialized co-processor card that's designed to handle them. You'll gain CPU independence and multiple high-speed

Tools that get the job done

We have off-the-shelf solutions for your common data transmission needs, and tools for custom situations. Select from HDLC/SDLC, Bisync, Async, X.25, LAP-B and custom protocols.

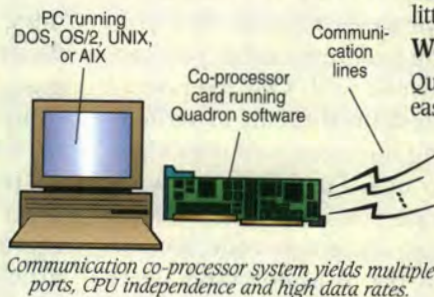
Your host PC can be an ISA (AT), EISA, or Micro Channel machine. You can run OS/2, DOS, AIX or UNIX. And perhaps best of all, you can port the co-processor code that you write from one operating system to another with little or no modification.

Work faster & simpler

Quadron software is easy to install, easy to use, and downright productive. Our high-level, C-language API makes your co-processor programming faster. The user manuals have numerous code examples to speed you on your way. And our support, should you need it, will serve up fast and accurate answers to your questions, whether they come in by phone, fax, or through our BBS.

Act right away

Does this sound like something you could benefit from? Contact us right now.

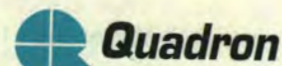


ports. Each port on a card can potentially support a different protocol and a different line speed.

Applications everywhere

Today, Quadron helps people manage demanding applications like:

- Travel reservation
- Stock market data delivery
- Telecommunications switching
- Retail point-of-sale purchases
- Process control
- Automatic teller services
- Shop floor control.



209 East Victoria Street
Santa Barbara, CA 93101 USA
fax 805-966-7630
phone 805-966-6424



© 1995 Quadron Service Corporation. All trademarks are the property of their owners.

Circle Reader Service Number 23



This article illustrates using the APL2 programming language. By DAVID LIEBTAG

Interactive PM Programming

OS/2's Presentation Manager is large, powerful, and complex. Since it has nearly 600 Win and Gpi services, learning how to use it is a daunting task. However, many of the difficulties in learning Presentation Manager can be overcome by taking an interactive approach.

When using a compiled language, the learning process requires that you have a working program before you can even start to work with OS/2 services. Obviously, it is a bit difficult to write a program using services you don't yet know how to use. Furthermore, with each new service you want to learn how to use, you must write new code, recompile, and then try to understand what went wrong if it doesn't work. These are expensive steps in an iterative learning process.

When using an interactive development environment, you can try a service and immediately see its effects. You can change the service's parameters and see what effects these changes cause. It is also easier to work with Presentation Manager messages in an interactive environment. Simply being able to examine the Presentation Manager environment at any time is a tremendous learning aid.

An interactive development envi-

ronment makes it convenient to explore questions like, I wonder what happens if I change the contents of the options in an accelerator table entry? It also makes it easier to explore Presentation Manager's less widely used but most powerful facilities, such as arbitrary unit presentation spaces and retained graphics.

APL2 is a general-purpose programming language used in a wide variety of disciplines. APL2 is an interpreted language and provides an interactive application development environment. APL2 for OS/2, also called APL2/2, includes an interface to Presentation Manager that can be used to interactively develop Presentation Manager applications. This article illustrates how you can easily exploit some of Presentation Manager's most powerful facilities using APL2/2.

THE APL2-PM INTERFACE

The APL2/2 product's several components include a language interpreter, a session manager, and an auxiliary processor, called AP145, that provides an interface to Presentation Manager. The session manager provides a window through which you communicate with the language interpreter and AP145. APL2 components communicate by using a special class of variables called Shared Variables. A shared vari-



able is a piece of data that is common to or shared between two components.

A separate Presentation Manager environment is managed for each variable you share with AP145. Each environment includes a separate thread and message queue. AP145 monitors these queues and dispatches their messages. Most messages are immediately dispatched to the appropriate default window procedure. Only messages specifically requested by the application are processed in APL2.

AP145 supports many OS/2 services, including most of the Ddf, Dev, Dos, Drg, Gpi, Prf, Spl, and Win services.

To call a service, assign the shared variable an array containing the service name and any parameters required by the service. To retrieve the service's return code, you reference the shared variable. Here is an example of calling a Win service and retrieving the service's results:

```
SV145~'WinShowWindow' HANDLE TRUE
(APRC OSRC CMD)-SV145
```

Note: In APL2, a left-arrow symbol assigns the value on the right to the variable name(s) on the left.

For comparison purposes, here is the analogous call in C:

```
Rc = WinShowWindow(HANDLE,TRUE) ;
```

The APL2 and C methods both use the same Presentation Manager constants and service names that are documented in the Presentation Manager reference manuals.

The differences are that APL2 allows you to call the service interactively, and it will run asynchronously. You can continue with other processing before referencing the shared variable.

Each reference of an AP145 shared variable returns a three-element array. The first element is an auxiliary proces-

sor return code. This code is zero unless an error is encountered, such as an incorrect number of parameters. The second element is the return code of the service. The third element is the service name and its parameters. Services can update their parameters, so the values may be different from those originally assigned.

CREATING A WINDOW

To demonstrate how AP145 can be used interactively, the following examples show how to create a simple window and draw some graphics. You can type these commands directly into the APL2/2 session manager and watch Presentation Manager process the service calls.

First, share a variable with AP145 to establish a connection with Presentation Manager. The SVOFFER routine can be defined using an APL2 COPY command.

```
)COPY 1 UTILITY SVOFFER
145 SVOFFER 'SV145'
```

The first service to call is WinRegisterClass to register a class for a window that you can work with. The CS_SIZEREDRAW variable, and the other Presentation Manager constants used in this article, can be set using a COPY command.

```
)COPY 2 PMWIN CS_SIZEREDRAW
CLASS~'APL2 Client Class'
STYLE~CS_SIZEREDRAW
SV145~'WinRegisterClass' 0 CLASS 0 STYLE 0
```

Now, call WinCreateStdWindow to create a standard frame window, as shown in Figure 1; for the client class, use the class name you just registered. This will cause a standard window to appear on your desktop.

The FCF_SHELLPOSITION flag causes your window to be created with a standard position and size. However, it does



YOUR *single* SOURCE FOR OS/2 APPLICATIONS

UTILITIES PRODUCTIVITY SOFTWARE
BOOKS & VIDEOS GRAPHICS & CAD
OS/2 T-SHIRTS COMMUNICATIONS
BACKUP SOFTWARE NETWORKING
DEVELOPMENT TOOLS WORD PROCESSING

ORDERS: 1-800-776-8284

FAX: 919-878-7479

INQUIRIES: 919-878-9700

3209 Gresham Lake Road, Suite 135 Raleigh, NC 27615



FULL COLOR
CATALOG
NOW AVAILABLE



Circle Reader Service Number 24

TCP/2™

TCP/IP for OS/2®

Supports *all* released versions of OS/2
including 32bit API for OS/2 2.0 and above

**Our customers
say our tcp/ip is the best.
It's faster, more reliable,
and more complete.
Find out for yourself!**

A DAN LANCIANI PRODUCT

For further information contact: Essex Systems, Inc.
One Central Street • Middleton, MA 01949

(508) 750-6200

TCP/2 is a trademark of DLD consulting. Ethernet is a trademark of Xerox Corporation.
OS/2 is a registered trademark of International Business Machines Corporation. VT102 is
registered trademark of Digital Equipment Corporation. TCP/2 is based in part on
work done by the University of California Berkeley.

Circle Reader Service Number 25

The easy to use 32 bit OS/2 PM IPF Language Editor...

IPF Editor

- Add help to applications in minutes
- Designed for easy use by programmers and non-programmers
- Generate C and Resource source files to add help to applications automatically
- Import wordprocessing files, including WordPerfect, quickly and accurately
- Preview IPF output without compiling
- Create all hypertext links automatically
- Optional Voice Support Module
- Includes complete on-line help and documentation

IPF Editor: \$150
Voice Support Module: \$50

Orders:
1-800-IPF-7622

Information: (206)428-5025
on compuserve at
70410,2416

Perez Computing Services
4725 Monte Vista Place
Mt Vernon, WA 98273

Circle Reader Service Number 26

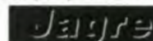
The Next Wave of Lotus Notes Applications Is Here...

Jagre, Inc. provides you with a whole new way to increase your daily work productivity using IBM OS/2-based desktop applications and Lotus Notes!

Welcome to Jagre SmartLink, the easiest quickest way to enable your OS/2-based desktop applications to work with Notes!

Jagre SmartLink is a set of nine Notes-enabled tools grouped together to provide you with increased productivity for Document Management, E-mail, Fax, Paging, OCR, Audio, Video and User-based Agents.

For more information about SmartLink, call Jagre at (617) 424-8302 or write to 102 Gainsborough St. #202E Boston, MA 02115



Circle Reader Service Number 27

FAX for OS/2

From the developers of *FaxWorks OS/2* and *PMfax*
Client/Server API and Printer Driver Toolkits
Support for LANs, up to 96 lines per CPU,
and all popular fax hardware

Keller Group Inc.

Voice: (612) 429-7273

CIS: 72120,3404

Faxworks is a trademark of SofNet, Inc.

Fax: (800) 329-3293

Fax: (612) 653-1987

PMfax is a trademark of Keller Group

Circle Reader Service Number 28

Opt-Tech Sort/Merge

High Performance Sort/Merge/Select Utility

Use as a stand-alone utility or call as a Subroutine. Many features including unlimited file size, multiple keys, record selection, duplicate elimination, summing and more!

Call for free brochure

Opt-TechData Processing
P.O. Box 678

Available for:

OS/2 or Unix \$249

DOS or Windows \$149

Zephyr Cove, NV 89448

Phone (702) 588-3737

Fax (702) 588-7576

Circle Reader Service Number 29

The First Twain Enabled Image Scanning & Viewing Utilities for OS/2

ATTALUSE

FEATURES:

Fast Image viewing • Easy Screen Capturing • Quick File Conversion • Efficient Color Modification • Powerful Image Manipulation • Real-Time Image Navigation

CURRENTLY SUPPORTS:

Logitech Scanman 256 • Hewlett Packard IIP • Hewlett Packard IIC Series

After The *ATTALUSE* Comes The *REVIEW*

Document Management System

Flexible & Fast Search/Retrieval • Multiple search Fields • Network Support and more...



Solution Technology, Inc. 1101 South Rogers Circle, Bldg 14 Boca Raton, FL 33487
Tel: (407) 241-3210 Fax: (407) 997-6518

Applause and Review are registered trademarks of Solution Technology Inc. All other brand names are trademarks of their respective owners. ©1994 Solution Technology, Inc.



Circle Reader Service Number 30


```

API-'WinCreateStdWindow'
PARENT-HWND_DESKTOP
STYLE-WS_VISIBLE
FLAGS-FCF_TITLEBAR + FCF_SYSMENU
FLAGS-FLAGS + FCF_SHELLPOSITION + FCF_SIZEBORDER
TITLE-'Interactive Window'
(CSTYLE DLL ID CLIENT)-0 0 0 0
SV145-API PARENT STYLE FLAGS CLASS TITLE CSTYLE DLL ID CLIENT
(APRC FRAME CMD)-SV145
(API PARENT STYLE FLAGS CLASS TITLE CSTYLE DLL ID CLIENT)-CMD

```

Figure 1. Calling *WinCreateStdWindow* to create a standard frame window.

not automatically place your window on the top of all the other windows on your desktop. To bring it to the top, use the *WinSetWindowPos* service:

```

FLAGS-SWP_SHOW + SWP_ACTIVATE + SWP_ZORDER
SV145-'WinSetWindowPos' FRAME HWND_TOP 0 0 0 0 FLAGS

```

If you are trying these examples as you read, you will see that the client area of the window you just created is transparent because you have not yet drawn any graphics in it. To make it easier to see the effects of your calls to Gpi services, you may want to use your mouse to reposition the window now.

To draw in any window, you need a device context and a presentation space. A device context is a connection to a particular device like a display. A presentation space is a virtual coordinate space associated with a device context. You can draw graphics and text in a presentation space without worrying about device dependencies. Presentation Manager manages the conversions from the presentation space to the physical device.

Creating a device context can be quite complicated, but Presentation Manager provides a very simple service that will open a device context that is already associated with your window. It is called *WinOpenWindowDC*:

```

SV145-'WinOpenWindowDC' CLIENT
(APRC DC CMD)-SV145

```

Now you can create a presentation space. To make the presentation space easier to work with, you can define an arbitrary coordinate space. The example defines a range of 0 to 1000 in both the X and Y directions:

```

PSSIZE-SIZEL CAST 1001 1001
FLAGS-PU_ARBITRARY + GPIA_ASSOC
SV145-'GpiCreatePS' 0 DC PSSIZE FLAGS

```

```
(APRC PS CMD)-SV145
```

The presentation space now exists and you can draw on it, but by default, it is mapped to the entire display screen. To have your space mapped to the client window, you must set the page viewport to match the client's size and position. To do this, query the rectangle surrounding the client and set the page viewport to match this rectangle:

```

RECTANGLE-RECTL CAST 0 0 0 0
SV145-'WinQueryWindowRect' CLIENT RECTANGLE
(APRC OSRC CMD)-SV145
(API CLIENT RECTANGLE)-CMD
SV145-'GpiSetPageViewport' PS RECTANGLE

```

Now you're ready to actually draw in your client window. You can use all the Gpi services and your 0 to 1000 coordinate system. Presentation Manager will clip and scale your graphics to the client area.

To start, try setting the color and drawing a box that fills the entire client area:

```

SV145-'GpiSetColor' PS CLR_BLUE
POSITION-POINTL CAST 0 0
SV145-'GpiMove' PS POSITION
CORNER-POINTL CAST 1000 1000
SV145-'GpiBox' PS DRO_OUTLINEFILL CORNER 0 0

```

Now try changing the color and drawing a smaller box with rounded corners:

```

SV145-'GpiSetColor' PS CLR_RED
POSITION-POINTL CAST 100 100
SV145-'GpiMove' PS POSITION
CORNER-POINTL CAST 900 900
SV145-'GpiBox' PS DRO_OUTLINEFILL CORNER 50 50

```

Next, change the color once more and draw some text:



```
SV145-‘GpiSetColor’ PS CLR_GREEN
POSITION-POINTL CAST 450 500
SV145-‘GpiMove’ PS POSITION
SV145-‘GpiCharString’ PS 11 ‘Sample Text’
```

Using your presentation space, you can try some of the other Gpi services. GpiErase, GpiLine, and GpiMarker are all useful and easy services you might try.

As you can see, you can accomplish a lot calling OS/2 services interactively. You can create windows and experiment with the Presentation Manager services and structures. Eventually, however, you need to respond to user input; you need to process Presentation Manager messages. Fortunately, this too can be done interactively.

PROCESSING PRESENTATION MANAGER MESSAGES

APL145 provides several ‘Apl’ services that are used to redirect messages to APL2 for processing. The procedure is to tell AP145 what messages you want to process and then wait for them.

Your example window needs to respond to three events: size, paint, and close. When a size change occurs, you need to reset your page viewport so your coordinate system is mapped to the new window size. When part of your window has been invalidated, you need to repaint the invalid area. And when the user requests that the window be closed,

you need to close and destroy it. The following statements inform AP145 that you want the messages corresponding to these events to be passed to APL2 for processing:

```
SV145-‘AplRegisterMsg’ CLIENT WM_PSIZE 0 0
SV145-‘AplRegisterMsg’ CLIENT WM_PAINT 0 0
SV145-‘AplRegisterMsg’ FRAME WM_SYSCOMMAND SC_CLOSE 0
```

Once the messages you’re interested in have been registered, you use the AplWaitMsg service to wait for messages. AplWaitMsg updates its four parameters with the handle, message identifier, and message parameters 1 and 2:

```
SV145-‘AplWaitMsg’ 0 0 0 0
(APRC DSRC CMD)-SV145
(SERVICE HANDLE MSG MP1 MP2)-CMD
```

The last expression in the previous example will extract the window handle, message identifier, and the two message parameters from the result returned by AP145. You can compare the message identifier with any of the standard message identifiers to determine which message has been returned. For example, the following line of code yields a vector of Boolean values, which indicates that the message is a WM_SYSCOMMAND message:

```
MSG=WM_PSIZE WM_SYSCOMMAND WM_PAINT
0 1 0
```

When you receive a WM_PSIZE message, the size of your window has been changed. For your coordinate system to continue to work, you should query the client window rectangle and set the page viewport just as you did initially.

Presentation Manager sends you a WM_PAINT message when part of your window has been invalidated and needs to be redrawn. This can happen when part of your window has been covered and then exposed by movement of another window. In response to a WM_PAINT message, you should redraw the invalid area and tell Presentation Manager that the invalid part of your window is now valid so it does not keep sending you WM_PAINT messages.

To redraw the graphics, you can simply issue all the graphics calls again. Alternatively, you can use retained graphics to have the same effect more easily and quickly.

First, set the drawing mode to retained so you can draw graphics once and reuse them.

```
SV145-‘GpiSetDrawingMode’ PS DM_DRAWANDRETAIN
```

Casting APL2 Arrays

APL2 arrays are stored in an internal format that is different than Presentation Manager uses. Before using an APL2 array as an API parameter, the array must be converted to Presentation Manager’s format.

A program is supplied with APL2, which can be used to convert arrays from APL2’s to Presentation Manager’s formats. Here is how to make it available to your session:

```
3 11 □NA ‘CAST ATR’
```

The following shows how to build the variables you will need to cast the arrays used in this article:

```
LONG←‘I4 0 ’
POINTL←‘GO 1 2 ’,LONG,LONG
RECTL←‘GO 1 4 ’,LONG,LONG,LONG,LONG
SIZEL←‘GO 1 2 ’,LONG,LONG
```


Now open a segment, redraw your graphics as you did before, and then close the segment. When using retained graphics, once a segment contains graphics, you can draw the segment at any time and all the graphics will be redrawn. Following are the calls you use to open, close, and draw segments:

```
SEGMENT-1
SV145-'GpiOpenSegment' PS SEGMENT
SV145-'GpiCloseSegment' PS
SV145-'GpiDrawSegment' PS SEGMENT
```

Now you're ready to efficiently handle WM_PPAINT messages. Here's some code to query the invalid rectangle, tell Presentation Manager it's valid, and draw your segment:

```
RECTANGLE-RECTL CAST 0 0 0 0
SV145-'WinQueryUpdateRect' CLIENT RECTANGLE
(APRC OSRC CMD)-SV145
(API CLIENT RECTANGLE)-CMD
SV145-'WinValidateRect' CLIENT RECTANGLE TRUE
SV145-'GpiDrawSegment' PS SEGMENT
```

Finally, you also need to respond to WM_SYSCOMMAND messages. You can verify that the message actually corresponds to a close request from the system menu by examining the first message parameter. When you're done, you can simply destroy your window:

```
MP1=SC_CLOSE
1
SV145-'WinDestroyWindow' FRAME
```

When `AplWaitMsg` returns a message, you can call as many Presentation Manager services as you want and even continue to interact with your window before returning a message result. Messages that are re-

ceived while you are processing a message are held until the next call to `AplWaitMsg`. Once you are done processing the message, you can use `AplReturn` to return a result or you can use `AplWaitMsg` to return a result and wait for further messages.

SUMMARY

In this article you've seen how to build a simple window that uses some of the OS/2 Presentation Manager's most sophisticated facilities, including arbitrary unit presentation spaces and retained graphics. Using an interactive approach gives you a convenient way to explore Presentation Manager's many services, and APL2 is an excellent tool to do this.

David Liebttag, IBM Corp., San Jose, Calif., is an advisory programmer in IBM's APL Products and Services group. David's recent work has focused on the Presentation Manager facilities in the APL2 for OS/2 product. This work has included the session manager, editors, printing facilities, and the APL2-PM interface processor. David can be reached via Internet at liebttag@stlvm20.vnet.ibm.com or CompuServe at 73164,623.

REFERENCES

- OS/2 2.1 Library Presentation Manager Programming Reference Volume I (S10G-6264-01).
- OS/2 2.1 Library Presentation Manager Programming Reference Volume II (S10G-6265-01).
- OS/2 2.1 Library Presentation Manager Programming Reference Volume III (S10G-6272-01).



New Products for OS/2

RexxBase 1.34. This is an OS/2 dynamic link library (DLL) that allows REXX command procedures to create databases using dBase file formats. File allocation table (FAT) and High Performance File Systems (HPFS) are supported; RexxBase uses the REXX application program interface.

American Coders Ltd. Circle No. 151
Phone: (919) 846-2014

PowerPak for OS/2. This add-on utility is designed to complement OS/2 Warp 3. PowerPak has a full-featured program scheduler that enables users to launch various programs based on user-defined time intervals. The scheduler provides a clock and calendar for the OS/2 desktop. PowerPak's import and export facility helps users share data between OS/2 applications such as faxes, personal information managers, and databases.

Arcadia Technologies Inc. Circle No. 152
Phone: (818) 446-6945

Extra! for OS/2. This host connectivity software is 32 bit, object oriented, multithreaded, and compatible with Warp. Extra! for OS/2 supports advanced program-to-program communication and a wide range of productivity features, including integrated batch file transfer, macros, SmartPad, Hotspots, drag-and-drop keyboard, and color remappers.

Attachmate Corp. Circle No. 153
Phone: (206) 649-6551

Avista. This fully integrated, multiuser, client/server accounting and business management software solution runs on DOS, OS/2, or UNIX. Avista is based on a customizable relational database specifically designed to aggregate high-volume transactional data typical to

accounting systems. It is aimed to be high performance and still cost-effective.

Avista Software Circle No. 154
Phone: (404) 564-8015

BBN Internet Server. This UNIX server has the ease of use of BBN's graphical user interface-based software that runs on Apple Macintosh or Microsoft Windows desktops. The server supports all Internet-protocol client platforms, including Windows, Macintosh, UNIX, and OS/2. It provides access to widely used Internet server functions including World Wide Web (WWW) database servers, e-mail, Gopher database servers, FTP file retrieval servers, and Network News bulletin boards.

BBN Circle No. 155
Phone: (800) 632-7638 or (617) 873-8730

DataLink for Lotus Notes. This is a cost-effective, point-and-click solution for migrating and synchronizing data between Lotus Notes and other major relational databases. DataLink does not require scripting or programming. It offers complete server-platform independence and provides access to data stored in Lotus Notes 3.0 or higher servers running Windows, OS/2, UNIX, or NLM.

Brainstorm Technologies Circle No. 156
Phone: (617) 492-3399

Preditor/2. This graphical, 32-bit editor is equipped with the ability to display multiple windows, search across multiple files, edit and compile simultaneously, and emulate other editors. It is customizable—programmers can jump to variable declarations, class definitions, or start functions.

Compuware Corp. Circle No. 157
Phone: (810) 737-7596



VisualGen. This application development tool for business-critical applications allows programmers to develop client and server code concurrently for use on workstations and the host. Users can take advantage of cooperative processing. The company claims its product brings new functionality through support for Windows clients and C++ application generation for OS/2 and AIX.

IBM
Phone: (914) 766-1211

Circle No. 158

Visual SlickEdit for OS/2. This programmers' editor offers procedure tagging, multiple clipboards, and compiler error processing. It provides the user with a macro language called Slick-C as well as a DLL interface for extending the editor. Visual SlickEdit supports C/C++, Pascal, Fortran, Basic, dBASE, Modula-2, assembly language, COBOL, and Ada.

microEdge
Phone: (919) 790-1691

Circle No. 159

NDP Pentium Compilers. These compilers run under DOS and OS/2. They are available for Fortran, C/C++, and Pascal. They drive RISC, CISC, and Vector devices. The new compilers add Pentium scheduling, the use of FXCH instructions to streamline x87 stack accesses, and a new peepholer that results in smaller code. Other features are dynamic linking, loop unrolling, numeric register caching, and numeric register coloring.

Microway
Phone: (508) 746-7341

Circle No. 160

Wizard O.S. 1.0. This multi-operating-system boot utility allows users to choose between DOS, Windows 95, OS/2, Windows NT, and UNIX at boot time. The user can have multiple versions of DOS or OS/2 and thus have different configurations of a particular operating system. One hundred different operating systems can be installed on the same hard disk. It does not require any special disk partitions like Boot Manager, nor does it require any disk partitioning or formatting.

Modular Software
Systems
Phone: (800) 438-3930 or (206) 631-5781

Circle No. 161

Quick/2 Install. This installation tool allows the entire OS/2 operating system to be installed in less than 15 minutes using quarter-inch cartridge backup systems. Quick/2 is ideal for users who require the ability to perform several installations or who desire an "unattended" installation.

Parallel Storage Solutions
(914) 347-7044

Circle No. 162

ObjectPro. This is an object-oriented development tool for building robust client/server applications. The company announces that benefits include intuitive modeling, code reuse, and productive maintenance that applies to the whole business model—business logic as well as screens and database interaction. ObjectPro supports Oracle and Sybase through native interfaces. ODBC drivers are provided that support Ingres, SQLBase, DB2/2, Allbase, dBase, Access, FoxPro, and so on.

Trinzic Corp.
Phone: (617) 891-6500

Circle No. 163

GUI-Kit. This C and C++ cross-platform GUI development toolkit supports 32-bit development under Win32, Windows NT, OS/2, and UNIX/Motif. This object-oriented programming environment can be used in C or C++.

Visual Systems Corp.
Phone: (612) 434-6382

Circle No. 164

The Graham Utilities for OS/2. New utilities are added to the already strong suite of over 30 applications. The user can recover multiple deleted files in one step and recover files from damaged or unusable High Performance File System partitions. Two file defragmenters are included.

Warp Speed Computers
Phone: 613 384 1060

Circle No. 165

Mathematica 2.2. Wolfram Research announces the release of a native version of Mathematica 2.2. The new version takes advantage of OS/2's 32-bit environment and runs faster. It is compatible with OS/2 Warp 3.0.

Wolfram Research Inc.
Phone: (217) 398-0700

Circle No. 166



Buyer's Guide

The staff of OS/2 Developer put together this special section to guide you through the various software tools specifically for the performance tuning, testing, and debugging of OS/2 applications. The products described in this guide are based on surveys sent to vendors; they are provided as a service to you and are free to vendors. These listings do not represent an endorsement by OS/2 Developer. Although every effort has been made to ensure accuracy, we do not assume any responsibility for error or omission in this section.

Testing and Debugging Tools Buyer's Guide

AUTOTESTER

Circle No. 101

AutoTester for OS/2 2.02 is an automated testing tool designed specifically for OS/2 Program Manager applications. AutoTester builds structured, object-oriented, and documented tests as you operate an application. Thus, the user gets a flexible, reusable, and maintainable test library that can be executed unattended. It supports the skilled developer and expert application users. Price: \$5,000 per copy with quantity discounts.

AutoTester, 8150 N. Central Expressway, Ste. 1300, Dallas, Tex. 75206, (800) 328-1196 or (214) 368-1196, fax (214) 750-9668.

BON AMI

Circle No. 102

CPU Monitor Plus 2.31 is an OS/2 Performance Monitor and Analysis package. It displays real-time CPU, RAM, Disk, Interrupt, and COM port activity. It will diagnose poorly running systems, set execution priorities, and log performance data for later analysis. It can start, stop, or suspend programs and has over 100 different metrics and ratios. Price: \$129.95.

Bon Ami, 60 Thoreau St. Ste. 219, Concord, Mass. 01742, (508) 371-1997, fax (508) 371-2333.

CARRY ASSOCIATES

Circle No. 103

Sys Maint 3.3 is a 32-bit Presentation Manager application that provides a full desktop diagnostic and a backup and repair utility. It identifies and fixes a number of common Desktop

problems, provides a desktop backup and restore, allows the user to edit, copy, move, backup, search, compare, size, repair, and maintain any INI file, including the OS2SYS.INI and OS2.INI files. It will view, copy, move, split, join, test, compare, and maintain extended attributes, including the desktop directory structure. Price: \$49.95 plus \$7.00 shipping and handling.

Carry Associates, 990 Ironwood Ct., Marco Island, Fla. 33937-4458, (813) 642-9126, fax (813) 642-1007.

CLIENT SERVER

NETWORKING INC.

Circle No. 104

WATCHIT 2.0 optimizes IBM LAN platforms (OS/2, AIX, MVS/VM, and AS/400). WATCHIT collects statistics about resource and user activity and analyzes the data. It automates collection of IBM LAN server capacity and performance statistics to help improve performance and anticipate capacity problems. Collection may be automated at the server or manually controlled from your desktop. Price: \$349.

Client Server Networking Inc., P.O. Box 370111, West Hartford, Conn., 06137-0111, (203) 233-2951, fax (203) 233-2951.

IBM CORP.

Circle No. 105

Workstation Interactive Test Tool (WITT) 2.2.2 is a capture and playback tool for automating unit and regression testing. WITT records keystrokes, mouse movements, and screen images as you step through your application



to automatically generate test scripts. WITT can also be used to automate many activities such as giving demos, entering operator commands, starting and stopping subsystems, system testing, and simulating users for disaster recovery. WITT installs on the OS/2 desktop and allows testing of Host, OS/2 Presentation Manager, and Text applications.

IBM Corp., 555 Bailey Ave., San Jose, Calif. 95141, (800) 426-4785 or (408) 463-2000, fax (800) 426-4522.

MERCURY INTERACTIVE CORP. Circle No. 106

WinRunner for OS/2 is an automated testing tool for OS/2 applications that compresses manual testing into a single overnight run. It tests OS/2 applications in any programming language. WinRunner includes automatic GUI verifications, inside testing beyond the graphical user interface, output synchronization, text recognition, object-oriented record and replay, and reusable and portable tests. Price: \$2,850.

Mercury Interactive Corp., 470 Portrero Ave., Sunnyvale, Calif. 94086, (800) 837-8911 or (408) 523-9900, fax (408) 523-9911.

MICROQUILL SOFTWARE PUBLISHING INC.

Circle No. 107

SmartHeap 2.2 is a fast (3X-100X), portable, reliable ANSI-compliant, and malloc-free library. It is thread-safe and supports multiple memory pools and shared memory. SmartHeap provides heap error detection and detects bugs other tools miss—leakage, overwrites, double-frees, wild pointers, out of memory, and references to previously freed memory. Price: \$695.

MicroQuill Software Publishing Inc., 4900 25th Ave. N.E., Ste. 206, Seattle, Wash. 98105, (800) 441-7822 or (206) 525-8218, fax (206) 525-8309.

PERISCOPE COMPANY INC. Circle No. 108

Periscope/32 for OS/2 5.4 is a device driver

debugger for OS/2 v. 2.x. This full-screen, symbolic, source-level debugger helps developers of OS/2 2.0 Base and Presentation Manager drivers. It operates at the systems level, has easy access to any memory location in the system, and is compatible with applications level debuggers. It includes host and target system software and a break-out switch for crash recovery.

Periscope Company Inc., 1475 Peachtree St., Ste. 100, Atlanta, Ga. 30309, (800) 722-7006 or (404) 888-5335, fax (404) 888-5520.

PROGRAMART CORP.

Circle No. 109

APMPower 2.0 is an OS/2-based product that facilitates the analysis of MVS application performance data. Operating with Programart's STROBE application performance measurement system, APMPower is used to pinpoint and resolve application performance problems before they enter production. It enables users to freely share application performance knowledge, improving IS productivity and application quality. Price: \$2,000 for single user, \$4,000 for current-use licenses. Volume discounts available.

Programart Corp., 124 Mt. Auburn St., Cambridge, Mass. 02138, (617) 498-4045, fax (617) 868-6069.

SOFTBRIDGE INC.

Circle No. 110

Automated Test Facility 3.5 tests client/server and stand-alone OS/2, Windows, Windows NT, and legacy (via 3270 emulation) applications. ATF's two-tiered architecture supports true client/server testing. From one central point, ATF can run intertwined, simultaneous, unattended tests on multiple PCs, testing all client/server layers—graphical user interfaces, distributed applications, and legacy systems. Price: \$18,000 for base system.

Softbridge Inc., 125 Cambridge Park Dr., Cambridge, Mass. 02140, (800) 955-9190 or (617) 576-2257, fax (617) 864-7747.

Take control of your OS/2 scheduling needs with
Advanced Task Scheduler for OS/2

September						
		1	2	3		
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

ATS

Version 3.0



Are you looking for a way to manage your production job streams? Do you have programs that need to run after the completion of one or more other programs? Do you want to run programs on a regular basis? Do you want to process files that have just been received from another system or modified locally? Do you need to take action if a file is missing? Do you need to schedule around holidays and periods of heavy CPU load?

With ATS you can run your programs when YOU want them to run with out you having to be there.

Upgrades Available
from Version 2 and
other Task Schedulers
Call or E-Mail for Details

MHR
Software and Consulting
2227 U.S. Highway #1 * Suite 146
North Brunswick, NJ 08902

Voice: (908) 821 - 0359 * Fax: (908) 821 - 0350 * CompuServe 70312,627

Here are some of the features of
ATS for OS/2:

- * Manage Production Job Streams
- * Programs can be scheduled to run anytime.
- * Programs can be dependent upon Files, Other Programs, Holiday Schedules, and External Signals
- * Integrated Operators Console
- * Logging - To File and Console

New Features:

- * Job Queues - for managing your resources while managing your tasks
- * Periodic Scheduling
- * COMPLETE API and Command Line Interface

Only
\$349

Circle Reader Service Number 31

SOFTOUCH SYSTEMS INC. Circle No. 111
Gamma Tech Utilities 3.0 for OS/2 Warp accelerates disk performance, performs High Performance File System (HPFS) optimization on line, and defragments file allocation table (FAT) files without crosslinking their extended attributes. Functions include desktop backup, disk analysis, boot sector backup, restore, and virus protection. Price: \$99.

SofTouch Systems Inc., 1300 S. Meridian Ave. Ste. 600, Oklahoma City, Okla. 73108, (405) 947-8085, fax (405) 632-6537.

2500AD SOFTWARE INC. Circle No. 112
2500AD Simulator/Debugger 5 is a full-color, menu-driven debugger. It debugs C and assembly language source code. It features stack control, memory traps, and control of registers, flags, memory, and variables. Price: \$150.

2500AD Software Inc., 109 Brookdale Ave., Buena Vista, Colo. 81211, (800) 843-8144 or (719) 395-8683, fax (719) 395-8206

VERITAS SOFTWARE CORP. Circle No. 113
VistaTest supplies the basic tools needed to immediately improve detection and elimination of latent defects. VistaTEST provides quick insight into what part of mission-critical application code has not been exercised by existing test suites. Dynamic coverage includes predicate, conditional, block, and/or path levels of code coverage. The product supports the IBM C Set 2 and C Set ++ compilers, including templates and exception handling. Price: VistaTEST (C), \$1,800; VistaTEST (C/C++), \$2,800.

VistaSIM simulates interface behaviors. Using either prepackaged OS calls or customized libraries created with the VistaSim Simulation Builder, this functionality allows for testing of difficult-to-reach code. Price: \$3,200.

VistaGRAPH is a general-purpose graphing tool that provides presentation graphics of charts, line plots, and kiviatt diagrams. These charts provide clear visual representation of test results.

VERITAS Software Corp., 4800 Great America Parkway, Santa Clara, Calif. 95034, (800) 258-8649 or (408) 727-1222, fax (408) 562-4334.

VISUAL SlickEdit™

THE INTELLIGENT
PROGRAMMER'S
EDITOR

\$199

SPECIAL OS/2
INTRODUCTORY
OFFER!

OS/2
WINDOWS
WINDOWS NT

Winners of Software
Development Productivity Award

Visual SlickEdit™ is the high powered programmer's editor with the speed, ease, and features you need to ignite your productivity. Learn in no time with our comprehensive CUA, Brief, Emacs, and VI emulations. Take off with our extensive language support, project management, and NEW SmartPaste™.

Powerful features:

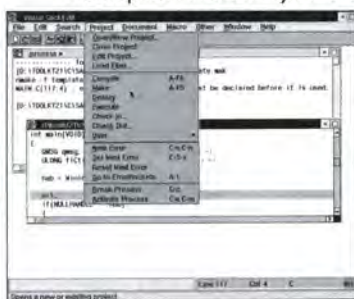
- Incremental search/search and replace
- Completely configurable
- Spell check comments and strings
- Typeless, object-oriented, C-style macro language
- Built-in dialog editor
- Clipboard Inheritance™ (patent pending)

Visual SlickEdit supports C/C++, Pascal, Fortran, Basic, dBASE, Modula-2, Assembly, COBOL, and Ada. The Windows NT version is available for Intel, MIPS, and Alpha AXP machines.

To Order, Call 800-934-EDIT also (919) 831-0600 or FAX (919) 831-0101

Visual SlickEdit, SmartPaste, and Clipboard Inheritance are trademarks of MicroEdge. Windows NT, MIPS, and Alpha are trademarks of their respective manufacturers.

MicroEdge, Inc. PO Box 18038, Raleigh, NC 27619-8038 USA

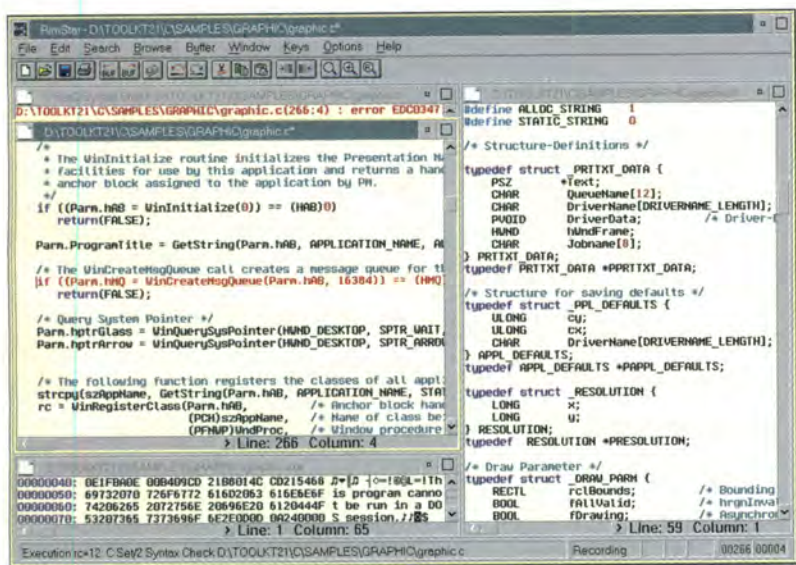


Cut your work in half using:

- Procedure tagging
- Syntax color-coding, expansion, and indenting
- SmartPaste reintends pasted source code according to nesting level
- Compiler error processing

Circle Reader Service Number 32

POWER UP WITH THE RIMSTARTM PROGRAMMER'S EDITOR NEW VERSION 2.1



If you're looking for a fully configurable, professional OS/2 PM programmer's editor to replace your current text-mode editor – we have what you have been looking for!

No hassle changing editors

We know changing editors can be a major hassle. You don't want to relearn a new set of keystrokes no matter how good the product. Well, you don't have to – we have Brief, CUA, Epsilon, Multi-Edit, SlickEdit, PWB, and Borland IDE keyboard mapping waiting for you. If you're not using one of these, let us know and we will create the mappings you need.

Features designed to increase productivity

It has all the features you have come to expect, plus special features designed to anticipate your needs and increase your productivity. Features like a 'C' source browser that eliminates those time consuming searches for functions, global variables, typedefs, and macros by providing you with instant positioning to both definitions and references. Using our powerful 'C' macro language you can customize your programming environment to suit your individual needs.

Partial Feature List

- ✓ Complete ANSI 'C' macro language
- ✓ SyntaColor[™]—syntax highlighting
- ✓ Smart C/C++ indenting & brace matching
- ✓ Bookmarks
- ✓ Unlimited undo and redo
- ✓ Timed auto save
- ✓ Access PDK help for function under cursor
- ✓ Multi-threaded for no waiting
- ✓ Compile and jump to errors
- ✓ Import/export to system clipboard
- ✓ Keystroke record/playback
- ✓ Block indent/outdent
- ✓ Hex editing
- ✓ Customizable menus
- ✓ Column, line and block selection, search and replace
- ✓ Integrates with Workframe/2
- ✓ Source browser for 'C'
- ✓ Template editing
- ✓ Multi-buffer regular expression search and replace
- ✓ Support for version control
- ✓ Complete on-line help
- ✓ Save and restore state between sessions
- ✓ OS/2 2.x 32 bit PM Multi-document interface

"My copy of Brief has been permanently retired. Keep up the good work!" - A.L.

RimStar Technology, Inc.

91 Halls Mill Road
Newfields, NH 03856
Voice: (603) 778-2500
Fax: (603) 778-2408
BBS: (603) 778-4644

Price \$299.00

Plus Shipping & Handling.

To order call 1-800-746-7007

60 day money-back guarantee.

Also available for Windows and Windows NT

Circle Reader Service Number 33

All products and company names are trademarks or registered trademarks of their respective holders. RimStar and SyntaColor are trademarks of RimStar Technology, Inc.

© 1994 RimStar Technology Inc.

The Suite Way to Build Portable Applications

NEW!

Is building applications your job? Then life just got easier thanks to the zApp® Developer's Suite for Windows. The zApp Developer's Suite is a set of highly integrated C++ development tools designed to help you transform the blueprints in your mind into commercial quality applications—quickly and easily. And best of all, applications built using the zApp Developer's Suite are portable to fourteen different platforms!

The zApp Developer's Suite consists of zApp, the award-winning portable C++ application framework; zApp Factory™, a fully visual application designer and code generator; and the zApp Interface Pack, a collection of high-level visual objects for the zApp environment. All of these tools are highly integrated to provide maximum ease of use and flexibility.

Rapid Application Development.

Introducing an exciting new visual technology that lets you drag and drop a wide assortment of objects like toolbars, tables, and 3D dialogs; define their characteristics; and build interfaces of any



complexity; all in one powerful but easy to use environment. With the click of a button, you can engage a powerful test mode which lets you interact with your application, seeing it exactly like your end user will see it: letting you fill in dialogs, pull down menus, etc. When you are pleased with the look and feel of your application, fully commented C++ source code is only a mouse click away, thanks to the zApp Developer's Suite's code generation capabilities.

Object-oriented Power.

The best news is that this development environment sits on top of zApp, the industry leading C++ application frame-

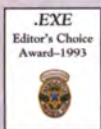
work, and the zApp Interface Pack, so you have all of their power at your disposal — toolbars, table objects, advanced graphics — in all, over 300 object classes of power just waiting to be tomorrow's best-selling application.

Portability and More.

When you're done building your application, *then* you can decide what platforms you want to support! Applications built using the zApp Developer's Suite are single-source portable to fourteen different platforms. By simply recompiling, your application will run natively on Windows, Win32 (Windows NT, Windows 95, and NT on the DEC Alpha), OS/2®, DOS Text, DOS Graphics and seven X/Motif platforms: IBM RS/6000 AIX, HP HPUX 9.x, SGI IRIX 5.2, SCO UNIX, Sun Solaris 2.x, Sun SunOS 4.1.x, Sun Solaris x86.

Free Demo.

Sound impossible? Well, if seeing is believing for you, call **1-800-346-6275**, ask for our free demo disk, and get a glimpse of what the future has to offer.



zApp and Inmark are registered trademarks of Inmark Development Corporation. zApp Factory is a trademark of Inmark Development Corporation. OS/2 is a registered trademark of IBM. All other trademarks are the property of their respective owners.

In the U.K. and Scandinavia, call PTS/Software Plus at +44 (0) 928 579900

In France, call PTS/Software Plus at (05) 908194

In Germany, call ESM Software at 07022-9256-0

In Italy, call Silicon Valley On-Line at (049) 654221

In Australia, call Micro Way at (03) 580-1333

BBS: 415-691-9990 • Internet: info@inmark.com
CompuServe: GO INMARK



Circle Reader Service Number 34

INMARK

2065 Landings Drive, Mountain View, CA 94043
T: 800-346-6275 or 415-691-9000 F: 415-691-9099

6362-0001-27

